RegressAR: Learning Object Pose based on Feature Points Local Information for Augmented Reality

Daniel M. Tokunaga* Escola Politécnica da USP
Munehiko Sato† MIT Media Lab
Dan Raviv‡ MIT Media Lab
Ramesh Raskar§ MIT Media Lab
Romero Tori¶ Escola Politécnica da USP

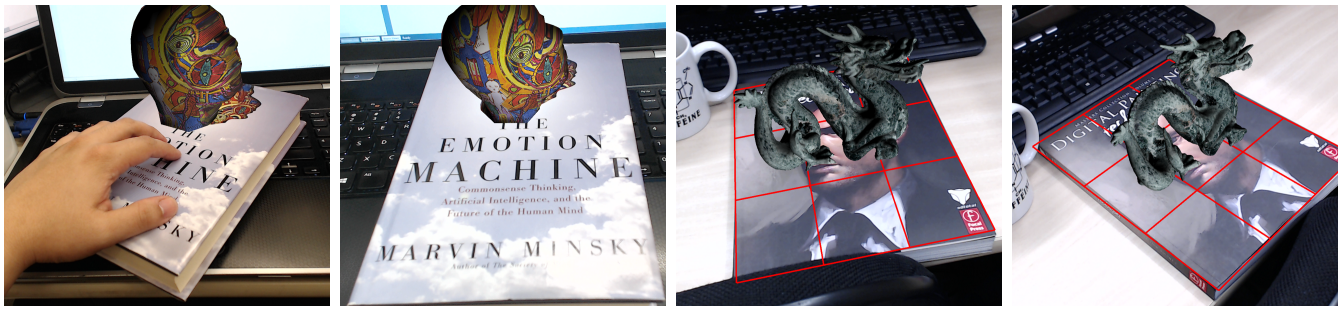


Figure 1: *Pose estimation based on our RegressAR approach.*

Abstract

In this paper we propose RegressAR, a novel approach, based on local poses of image feature points and decision forest regression, in order to fully estimate the object pose. By applying regression training over scale and rotation invariant feature points appearances, we obtain estimations of the local rotations. These local rotations are, then, projected to global object information in order to find the full pose. Unlike other works of pose regression by machine learning presented in the literature, we are the very first to fully achieve, by machine learning, image feature points local rotation; and full 6 degrees-of-freedom object pose. This solution also avoids complex non-linear solvers, applied in several state-of-the-art non-iterative solve Perspective-n-Point approaches. We show that this machine learning based approach, yet simple, can provide comparable accuracy to state-of-the-art solutions.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities I.4.8 [Image Processing and Computer Vision]: Scene Analysis—;

Keywords: Augmented Reality, Pose Estimation, Regression Forest

*e-mail:dmtokunaga@acm.org
†e-mail:munehiko@acm.org
‡e-mail:darav@mit.edu
§e-mail:raskar@media.mit.edu
¶e-mail:tori@acm.org

Links: DL PDF

1 Introduction

In the past years, a number of applications and ideas for augmented reality (AR) has been developed [Benko et al. 2014; van Krevelen and Poelman 2010]. However, linking a physical real objects with virtual information is still one central aspect on AR [van Krevelen and Poelman 2010]. Usage of real objects as link to virtual contents is not only limited to visual augmentation [Comport et al. 2006], but also with tactile sensations [Bau et al. 2012], as methods of input [Heun et al. 2013], or even output [Leithinger et al. 2014]. And one central issue related to the visual augmentation is the spatial registration of these objects, or in other words, the estimation of the objects poses.

Conventional methods for pose estimation with color images are often based on the correlation of known three-dimensional (3D) points to observed two-dimensional (2D) points. This correlation is known as perspective-n-points (P-n-P) problem [Fischler and Bolles 1981]. However, several solutions of this problem are based on complex non-linear equations solvers [Lepetit et al. 2009].

Also, in order to obtain this point correlation, feature points detection and matching [Lowe 1999; Bay et al. 2008] are applied in several P-n-P works [Lowe 1999]. These feature points can give rich information, which could be used at the estimation process, however in several P-n-P approaches, these information are thrown

away, where feature points are simplified as a single point in the space.

On the other hand, works such as presented by Fenzi et al. [Fenzi et al. 2013], explore machine learning regression approaches in order to learn the object pose, by the perspective distortion of these feature points; gathering more local information from each point. Despite that, they do not enable full pose estimation of the object.

In this work, we address this full object pose estimation with our novel full pose estimation approach, named RegressAR, which applies regression forest [Criminisi and Shotton 2013] over the feature points appearance changes, in order to locally learn the perspective changes. This regression can provide us local rotations of each feature point, rich information that can be used in several conditions, such as in deformation estimation [Tokunaga et al. 2015; Rendl et al. 2014]; that are not explored in conventional pose estimation works.

After that, we explore the prior information of the scene in order to refine the regression value returned by the regression forest combined with RANSAC method. Here, we focus on planar object pose estimation, evaluating our method in comparison to state-of-art P-n-P method. Yet, we also show that our methods could be extended to 3D rigid object or deformable objects, at the section 4.2.

Our main contribution in this work is our RegressAR approach itself, which can be a simple, yet accurate, alternative for AR systems. For the best of our knowledge, we are the very first to address this full pose of the object based on feature points local poses and machine learning approach. Thus, not needing to solve complex non-linear systems, replacing that with a fast and robust approach showing comparable results and dramatically lower real-time calculation demands.

In the next section we will present related works for object pose estimation. In section 3, we present our RegressAR approach itself. Our obtained results are presented in section 4, as well as its evaluation, in 4.1, and extensions, 4.2. Finally we discuss our approach and its results in section 5; and show our conclusion and future works in 6.

2 Related Works

Object pose estimation and tracking is one of key approaches in AR in order to obtain the spatial registration. This is achieved in several works by methods that solve the problem of Perspective-n-Points (PnP). Works such as [Lu et al. 2000; Lowe 1991] address this problem with iterative methods. Although these methods achieve good accuracy, they normally lack on computational performance [Lepetit et al. 2009].

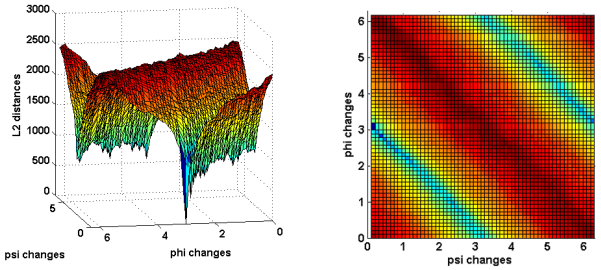
Works such as [Lepetit et al. 2009; Li et al. 2012] approach the solve P-n-P by noniterative methods. Li et al.[2012] address the PnP problem by minimizing several equations of 4th order polynomial, leading to solvers of 7th order, in order to obtain the rotation matrix first, to then obtain the translation matrix. Lepetit et al. [2009], on the other hand, address the PnP problem by expressing PnP to a weighted sum of 4 control points, and optimizing the 4th order polynomial using iterative Guass-Newton scheme. As we can see, all of these methods lay at the two points constrain in order to estimate the depth of the points, which, leads to complex non-linear polynomial solving problems, as pointed by [Kukelova et al. 2008].

On the other hand, machine learning based approaches had become largely explored in pose estimation of known objects. Those approaches have the advantage of avoiding complex non-linear solving problems. Murase and Nayar [1993] apply an eigen space analysis in order to learn the object poses and illumination changing. Then, works such as [Lepetit et al. 2009; Pepik et al. 2012; Gu and Ren 2010] also applied machine learning for training the object pose across its appearances. However, such works inherently deal with discretized views.

[Torki and Elgammal 2011; He et al. 2014; Hara and Chellappa 2014; Shimizu et al. 2015; Zhen et al. 2015] learn the object appearance across views in order to obtain continuous pose. Though, such approaches do not achieve full 6 DoF pose, neither the local pose of the feature points.

Besides that, works such as [Shotton et al. 2013; Brachmann et al. 2014; Tejani et al. 2014] also apply machine learning for full object pose estimation, despite based on depth and image information (RGB-D). In special, [Tejani et al. 2014] is similar to our approach, once authors train the local pose of the object, for each pixel. Also, all of those works apply random forest for pose estimation, as in our approach.

Partial local information of pose also can be estimated in [Fenzi et al. 2013], which apply regression based only on the feature points local appearance, and [Pepik et al. 2012], which apply deformable part model with HoG features. However, none of the works based on machine learning presented here achieve full 6 DoF object pose, nor local rotation. Although some works propose that full rotation can be achieved by simply extending its method; this extension is not easy, due to similarities in appearance of feature points such as presented in 2, specially to feature points that does not have rotation invariance.



(a) *3D view* **(b)** *top view*

Figure 2: *L2 distances between feature points views with fixed θ, and varying ϕ, ψ values of camera.* 0 *valued distance in the graph is the reference view. It is possible to observe the similarity which appears all across different ψ and ϕ, making the extension of 2DoF rotation regression (fixed ψ) to 3DoF rotation non-trivial.*

Our RF-RANSAC method (section 3.4) is also similar to [Brachmann et al. 2014], where regression forest method is combined with RANSAC method. Local feature points poses are also explored by [Tokunaga et al. 2015] for rigid and deformable object pose estimation. Despite those similarities, both works are based on RGB-D information, not being able to obtain local pose with only RGB images. Finally, [Wu et al. 2008] also shares similarities with our work, in the meaning that they also rely on local appearance changes for feature point matching; though, the final object pose is calculated with conventional RANSAC approach, also simplifying the local information to a simple point in space, and not achieving local rotation.

3 Pose Estimation Method

In this section we present our RegressAR approach for object pose estimation. The main idea behind our approach is to explore machine learning methods to learn the perspective changes based on

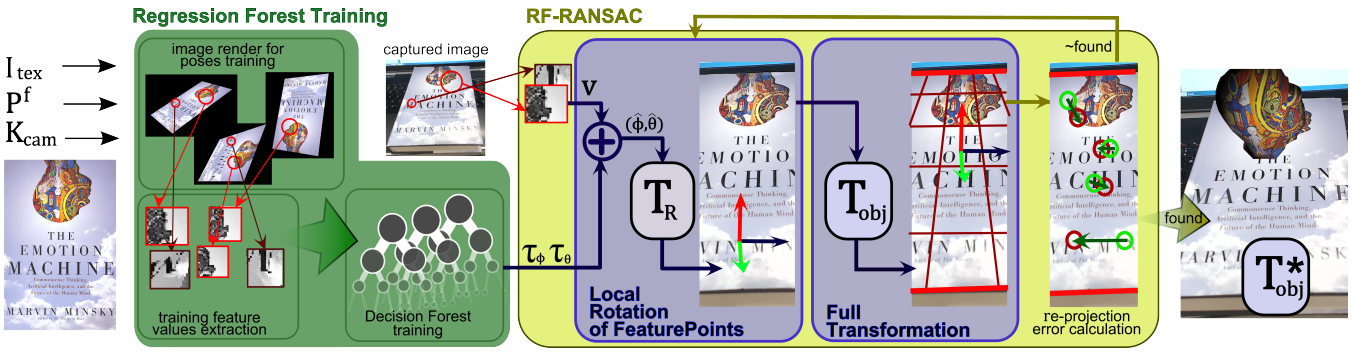


Figure 3: *Information flow of our approach.*

appearance changes around detected features in order to estimate the object pose. First, in order to achieve our pose estimation, we consider three different Euclidian coordinate spaces. The image space I, which are the two-dimensional space of captured images; and world coordinates w, the three-dimensional space of the scene where object pose needs to be calculated. At last, pre-captured information about the object are hold in o, the three-dimensional coordinate space of the original planar object model.

From this point of this work, we denote that variables with a uppercase I, such as x^I, denotes that are in image space; variables with o, the original coordinates of the captured object; and w, the new scene world coordinate. Also, points with upper case H denotes the homogeneous coordinates of the projected point, i.e., $\mathbf{p}^H = \mathbf{p}^w/z = [x/z, y/z, 1]^\top$.

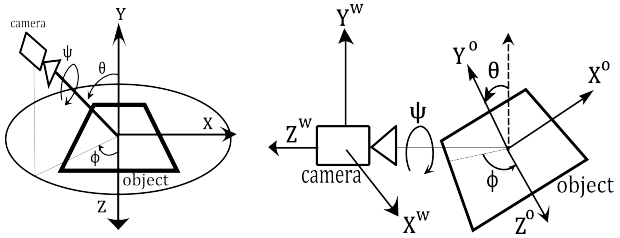
For perspectives changes training, we assume that the object is locally rigid and planar around the feature points. This assumption is always true for planar objects, but also could be extended for different types of object, such as 3D rigid objects [Lepetit et al. 2005] or locally rigid deformable objects, making our approach for these type of object as same as planar objects. Also, we assume that object surface texture I_{tex}, as well as, that the camera projection matrix $\mathbf{K}_{cam}$ are known.

Another assumption is that the θ and ϕ values of the camera are independent for the deformation of the projected surface to the image. This means that we can estimate each value independently of the other, as in [Fenzi et al. 2013]. Finally, we assume that the set of rotation and scale invariant feature points $P^f = \{p_0^f, \ldots, p_n^f\}$, such as SIFT [Lowe 1999] or SURF [Bay et al. 2008], spread across the object surface texture are known. Each feature point p_i^f contains a set of information $p_i^f = \{\mathbf{p}_i^o, \mathbf{o}_i^o, dsc_i\}$, where, $\mathbf{p}_i^o$ is the position of the point and $\mathbf{o}_i^o$, the orientation vector, both in the original object space; and dsc_i, the feature point descriptor.

Our camera model is similar to ASIFT work [Morel and Yu 2009], which can produce all the affine transformations for a planar object as presented by authors. Figure 4 presents our applied camera model. Note that ψ rotation is related to image space rotation. Furthermore, once feature points that we are applying for our method are rotation invariant [Bay et al. 2008], our training data can be ψ invariant, needing to train only θ and ϕ values per feature point deformations. At subsection 3.2, we show more details how to proceed this ψ-invariant training.

Our entire RegressAR approach is separated in four parts,

1. **regression forest training**: which generates the regression forest from samples of the object feature in several poses;
2. **local rotation pose estimation**: which estimates the rotation pose of the feature point based on the observed feature point local information;
3. **full transformation matrix estimation**: estimates the translation value of the feature points by using a second point;
4. **Random Forest Random Sample Consensus (RF-RANSAC)**: estimates the global object pose by combining decision forest and RANSAC approaches.



(a) *object centered coordinates* **(b)** *camera centered coordinates*

Figure 4: *Camera pose models for regression. Camera model in figure 4a is used in the training part, and figure 4b, for the pose estimation part.*

Figure 3 illustrates the entire approach information flow.

3.1 Regression Forest Training

The goal here is to train the appearance information around feature points across different θ and ϕ values for the camera position regression, based on the rotation invariant feature points. First, we use a synthetic approach to generate samples for training, similarly to [Morel and Yu 2009] and [Criminisi and Shotton 2013]. We render the object surface texture $\mathbf{I}_{tex}$ as planar object with different camera poses, using the same camera projection matrix $\mathbf{K}_{cam}$ of the scene capturing camera.

In each rendered image, we convert the color pixel values to brightness values and process a conventional feature matching algorithm [Bay et al. 2008] in order to find the matching feature points in the new image. Then, for each matched features point, we extract the feature point training feature values $\mathbf{v}_{tr}$.

This training features extraction is done by first extracting a small square image patch around the feature points. This image patch has a length of l_{diag}, which is based on the scale factor of the feature point, s_{ft}. Therefore, our transformation in the image space is extracted with size $l_{diag} = n_{pix} s_{ft} \sqrt{2}$. Where, n_{pix} is the number of the pixels to be extracted.

Then, this image patch is rotated using the inverse of orientation value o_{ft} of the feature point. This creates an image patch that always is oriented to the major orientation of the feature point, despite the ψ value of the frame. Then, this new patch is cropped to fit $l_{feat} = n_{pix} s_{ft}$, and rescaled to the size of the training samples feature l_{ft}.

This image patch is converted to an array of size l_{ft}^2, in order to obtain training feature values $\mathbf{v}_{tr} \in \mathbb{R}^{l_{ft}^2}$. At last, the feature values are normalized to a interval of $[0, 1]$, in order to make the feature less affected by brightness changes. Figure 5 illustrates this entire process.

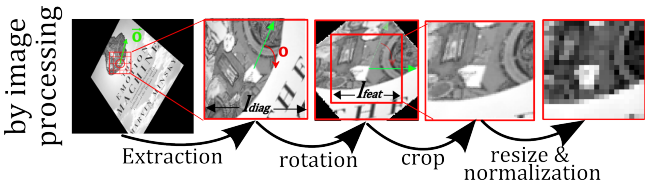


Figure 5: *Near feature point image patch extraction for training feature value* $\mathbf{v}$ *extraction by image processing.*

In order to achieve the well spread representative sampling data for the training, we generate several random samples with different θ values, within $[\pi/2, \ldots, 0]$, and ϕ, within $[0, \ldots, 2\pi]$. In order to train small errors created by rotation and scale values, we also use random ψ, x, y and z values. Furthermore, we add a gaussian noise of mean 0 and several standard deviations σs, in order to make our regression results more robust to image noises. Figure 6 illustrates part of the images and sample patch used at the training. Note that variations are more strong when θ is higher, once the image plane is less orthogonal to the camera.

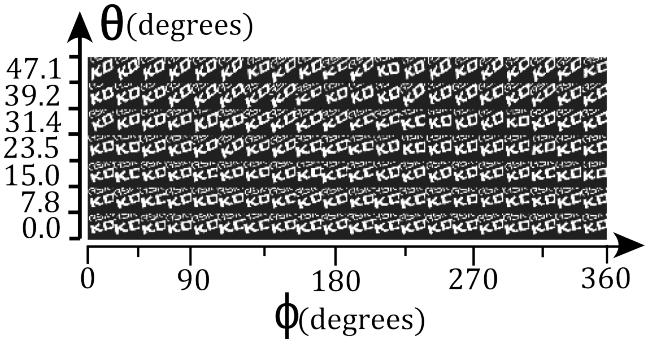


Figure 6: *Training feature values with different object poses across* θ *and* ϕ *values. Y axis shows changes in* θ *varying from* 90 *degrees to* 35 *degrees. X, across* ϕ *changes from* 0 *to* 360 *degrees.*

For the training itself, we apply a regression forest approach, due to its capability of generating fast regression results for real-time applications, among other advantages such as described in [Criminisi and Shotton 2013]. The decision forest approach is based on the multiple decision tree training[Criminisi and Shotton 2013]. Here, we generate n_τ decision trees for each extracted feature point f.

Once we assumed that θ and ϕ are independent, we can generate a set of n_τ trees, $\tau_{(\theta,f)}$, for the θ value, and n_ϕ trees, $\tau_{(\phi,f)}$ for ϕ values. However, decision forest also can be a memory expensive method, and training all the feature points of the object can rapidly increase the memory consumption. In order to avoid this problem, we can train only a set of P^{rf} perspective robust feature point within P^f. Here, we select P^{rf} as the n_{rf} feature points that most appear across different rendered images.

Other feature points within P^f, are still used as verification points at the calculation process, as described in section 3.4. At the end of this point, we have generated several feature points P^{rf}, each, attached with several regression trees $\tau_{(\theta,f)}$, $\tau_{(\phi,f)}$, where $P^{rf} \in P^f$, based the samples images.

3.2 Local Rotation Matrix of Feature Points

In order to process the pose estimation itself in a newly observed frame I_{obs}, we first proceed a feature matching algorithm to find the matching feature points, $\mathbf{P}_m^I$, between p_{obs}^I and P_f. Then, we capture the image patch as in section 3.1, for each matched point f within P_{rf}, points that has a regression forest associated.

We then extract the feature values vector $\mathbf{v}_f^t$, of each image patch. By applying a regression with the forest and feature vector, each regression tree will return a prediction of $\hat{\theta}$ and $\hat{\phi}$. Based on the returned values of $\hat{\theta}$ and $\hat{\phi}$, we first calculate the rotation matrix.

This rotation transformation is calculated as

$$\mathbf{T}_r = \mathbf{T}_\psi \cdot \mathbf{T}_\phi \cdot \mathbf{T}_\theta, \tag{1}$$

where $\mathbf{T}_\theta$ is a rotation around the x-axis with θ; $\mathbf{T}_\phi$, around y-axis with ϕ and finally, $\mathbf{T}_\psi$, a rotation around the direction of the camera, $(-z)$-axis. However, once the regression is ψ-invariant, now we need to re-estimate the ψ value for this new frame. This can be done by using a second known vector $\vec{v}_\psi$.

This $\vec{v}_\psi$ can be calculated by a second matched 2D point, or also by the extracted orientation o of the feature point. Using the orientation of the feature point itself enable us to calculate the rotation matrix of the object by only information from the feature point. This characteristic enable us obtain other unique results, as discussed at section 5. When using the orientation of the feature point itself, we set $\vec{v}_\psi^o = \vec{o}_f^o$ and $\vec{v}_\psi^I = \vec{o}_f^I$, where $\vec{o}_f^I$ is the observed vector calculated by orientation o returned by the feature point, $\vec{o}_f^I = [cos(o), sin(o)]$.

However, small errors can occur at the feature point orientation. These errors can lead in global pose error of the object. Using a second point as $\vec{\mathbf{v}}_\psi$, gives us more stable results. In this case, we set $\vec{v}_\psi^o$ to be the unit vector with same direction as $\mathbf{p}_f^o - \mathbf{p}_2^o$ and $\vec{v}_\psi^I$ unit vector with direction $p_f^I - p_2^I$, where $\mathbf{p}_2^o$ is the original 3D location of the feature point and p_2^2, the observed location of the second point at the image space.

After calculating $\vec{v}_\psi^I$ and $\vec{v}_\psi^o$, we calculate the final rotation matrix by finding $\hat{\psi}$ that matches the captured $\vec{v}_\psi^I$ with transformed $\vec{v}_\psi^o$, as illustrated by figure 7. Initially, $\vec{\mathbf{v}}^w(\hat{\psi})$ is calculated to be the final position of the transformed $\vec{\mathbf{v}}^o$ as

$$\vec{\mathbf{v}}^w(\hat{\psi}) = \mathbf{T}(\hat{\psi})\mathbf{T}_{\hat{\phi}}\mathbf{T}_{\hat{\theta}} \cdot \vec{\mathbf{v}}_\psi^o. \tag{2}$$

Where, $\mathbf{T}_{\hat{\phi}}$ and $\mathbf{T}_{\hat{\theta}}$ are the transformation matrices calculated by returned $\hat{\theta}$ and $\hat{\phi}$, and $\mathbf{T}(\hat{\psi})$ calculated with the unknown variable $\hat{\psi}$.

Once $\vec{\mathbf{v}}^w(\hat{\psi})$ projected to the image plane, defined by $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ axis, has same direction as $\vec{\mathbf{v}}_\psi^{obs}$. Where, $\vec{v}_\psi^{obs}$ is observed $\vec{v}_\psi^I$, projected to the world coordinates as

$$\vec{\mathbf{v}}_\psi^{obs} = (z_i \mathbf{K}_{cam}^{-1} \mathbf{p}^o - z_i \mathbf{K}_{cam}^{-1}(\mathbf{p}^o + \vec{\mathbf{v}}_\psi^I)). \tag{3}$$

Once, they must have same direction, we can calculate $\hat{\psi}$ by

$$\frac{\vec{\mathbf{v}}_\psi^{obs}[x]}{\vec{\mathbf{v}}_\psi^{obs}[y]} = \frac{\hat{\mathbf{X}} \cdot \vec{\mathbf{v}}^w(\hat{\psi})}{\hat{\mathbf{Y}} \cdot \vec{\mathbf{v}}^w(\hat{\psi})}. \tag{4}$$

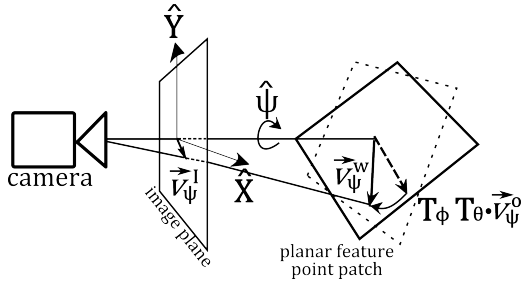


Figure 7: *ψ calculation model.*

By (4), we can find $\hat{\psi}$, that rotates the feature point to the same direction to $\vec{v}^I_\psi$. Therefore, being able to calculate the rotation matrix of the local feature point $\mathbf{T}_r$ by (1).

3.3 Full Transformation Matrix Estimation

After calculating the rotation matrix, we now estimate the full transformation matrix of each feature point. In order to estimate the full transformation of the object, we need to estimate the z position of the feature point. This is done by, again, applying the information of a second point $\mathbf{p}_2$. This calculation of z is done by fitting the distance between two rotated point to the observed distance.

First, we consider that the global position of a second point of the object $\mathbf{p}^w_2$, can be calculated by the sum of the position of the first observed feature point $\mathbf{p}^w_f$ and the rotated vector between both original positions $\mathbf{p}^o_2$ and $\mathbf{p}^o_f$. As

$$\mathbf{p}^w_2 = \mathbf{p}^w_f + \mathbf{T}_r \cdot (\mathbf{p}^o_2 - \mathbf{p}^o_f), \tag{5}$$

where, the global position of the observed feature point can be estimated by

$$\mathbf{p}^w_f = z^w \mathbf{K}^{-1}_{cam} p^I_f. \tag{6}$$

Note that here, the only unknown variable is z^w, z coordinate of $\mathbf{p}^w_f$. Also, note that this is true only for pair of points with rigid transformation, where the distance between both point does not change across time.

Then, by combining (5) and (6), and projecting the second point to the image plane we have

$$p^I_2 = \mathbf{K}_{cam}(z^w \mathbf{K}^{-1}_{cam} \cdot p^I_1 + \mathbf{T}_r(\mathbf{p}^o_2 - \mathbf{p}^o_1))^H, \tag{7}$$

where p^I_2 is the image coordinates of the observed second point. By (7), we can calculate the z^w coordinate of the observed feature point global position. Finally object translation is given by $\mathbf{t} = z^w K^{-1}_{cam} p^I_1 - \mathbf{T}_r \mathbf{p}^o_1$. By combining the rotation transformation obtained in section 3.2 and the translation matrix $\mathbf{T}_t$ calculated using $\mathbf{t}$, we obtain the object full transformation $\mathbf{T}_{obj}$ by

$$\mathbf{T}_{obj} = \mathbf{T}_t \mathbf{T}_r. \tag{8}$$

3.4 Random Forest Random Sample Consensus

For rigid object, one simple way to estimate the final object pose, based on the regression returned by several trees of several feature points, could be done by only taking the mean of returned $\hat{\theta}$ and $\hat{\phi}$, as conventional approaches [Criminisi and Shotton 2013], and using the same calculation of section 3.2 and 3.3.

However, as illustrated at figure 8, regressions returned by part of the trees has good accuracy, yet, other part has big errors around some values. In the first row of figure 8, we present L1 norm errors returned by 20 trees from one features values input. We can observe that, although θ returns good accuracy, ϕ regression has part of its trees with great errors.

This result is confirmed by the confusion matrices at the second row of figure 8. θ regression has a fall down in higher values, but this is caused by the fact that the image plane is less orthogonal to the camera direction. However, we also can observe that ϕ almost always have a second high probability peak, which creates wrong regression results.

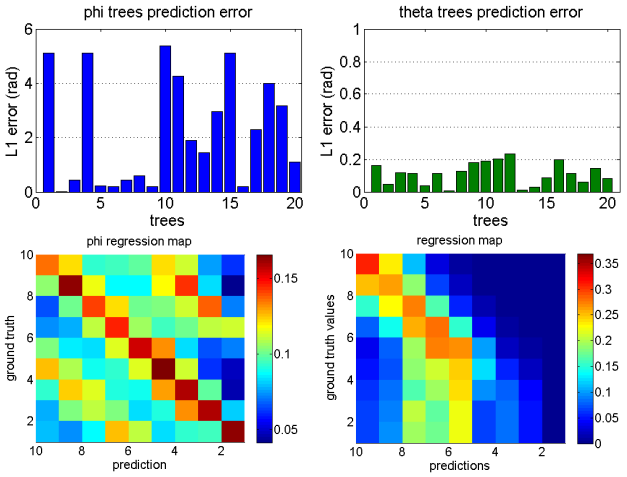


Figure 8: *Trees regression accuracy evaluation. First row: Per tree returned value L1 error for one value. Showing that part of trees return accurate results; however others, return wrong values around certain value. Second row: confusion matrix evaluation with 1000 non-trained images; Y axis represents the ground truth value, and X axis, the predicted value. Also illustrating the errors in ϕ regression.*

This could happen because of similarities of the patch deformations in two different views. However, unlike conventional decision trees, where the regression value is calculated based on the input feature values $\mathbf{v}$ as

$$p(\mathbf{T}_{obj}|\mathbf{v}) = \prod_i p_i(\mathbf{T}_{obj}|\mathbf{v}), \tag{9}$$

here in this work, we have also other prior information about the object pose that could be used in order to refine the pose regression.

One strong prior information are the distances between observed and reprojected 2D points of the feature points $\mathbf{p}^I_m = [(p^I_0, \mathbf{p}^f_j), \ldots, (p^I_n, \mathbf{p}^f_k)]$, such as used in many works of solve P-n-P problems [Lepetit et al. 2009; Li et al. 2012]. These information also can be applied to the forest regression in order to improve the overall accuracy.

We inject this prior information at (9) by calculating

$$p(\mathbf{T}_{obj}|\mathbf{v}, \mathbf{p}^I_m) = \prod_i p_i(\mathbf{T}_{obj}|\mathbf{v}, \mathbf{p}^I_m). \tag{10}$$

By applying the Bayes rule, (10) can be extended as

$$p(\mathbf{T}_{obj}|\mathbf{v}, \mathbf{p}^I_m) = \prod_i \frac{p_i(\mathbf{T}_{obj}|\mathbf{v}) p_i(\mathbf{p}^I_m|\mathbf{T}_{obj}, \mathbf{v})}{p_i(\mathbf{p}^I_m)}. \tag{11}$$

Here $p_i(\mathbf{p}^I_m|\mathbf{T}_{obj}, \mathbf{v})$ is the probability of finding the projected $\mathbf{p}^o$ to the image plane in $\mathbf{p}^I_m$ position by applying $\mathbf{T}_{obj}$. Once the reprojected positions $\mathbf{p}^I_m$ are independent of the feature values $\mathbf{v}$ used in regression, $p_i(\mathbf{p}^I_m|\mathbf{T}_{obj}, \mathbf{v})$ can be simplified as $p_i(\mathbf{p}^I_m|\mathbf{T}_{obj})$.

This reprojection probability $p_i(\mathbf{p}_m^I|\mathbf{T}_{obj})$ is inversely related to the sum of two-dimensional distance errors ϵ^I between the reprojected point, transformed by $\mathbf{T}_{obj}$, and observed point. This error can be calculated as

$$\epsilon^I = ||p_i^I - \mathbf{K}_{cam}(\mathbf{T}_{obj}\mathbf{p}_i^o)^H||. \quad (12)$$

Hence, as the distance ϵ increases, the probability $p_i(\mathbf{p}_m^I|\mathbf{T}_{obj})$ of T_{obj} decreases; we can approximate $p_i(\mathbf{p}_m^I|\mathbf{T}_{obj})$ to be inversely proportional to ϵ^I.

We now can rewrite equation (11) to estimate the best transformation T_{obj}^* as

$$\mathbf{T}_{obj}^* = \arg\max\ p(\mathbf{T}(\theta,\phi)|\mathbf{v},\mathbf{p}_m^I). \quad (13)$$

$T(\theta,\phi)$ can be calculated by the $\hat{\theta}$ returned by the trees $\tau_{(\theta,f)}^j$; and $\hat{\phi}$, by $\tau_{(\phi,f)}^k$, both from feature point f. We, also consider that the probability of the returned transformation from the trees $p_i(\mathbf{T}_{obj}|\mathbf{v})$, and the observed points $p_i(\mathbf{p}_m^I)$, are constant.

Finally, we have that $p_i(\mathbf{p}_m^I|T(\theta_j,\phi_k))$ is inversely proportional to sum of the re-projection error ϵ, as in (12), thus having

$$\begin{aligned}\mathbf{T}_{obj}^* &\approx \arg\min_{i,j,k}\sum_l \epsilon_{i,j,k}^I \\ &\approx \arg\min_{i,j,k}\sum_l \|p_l^I - \mathbf{K}_{cam}(\mathbf{T}(\theta_{(j,i)},\phi_{(k,i)})p_l^o)^H\|.\end{aligned} \quad (14)$$

By (14), we can calculate the final $\mathbf{T}_{obj}^*$ as the transformation that has least reprojection error calculated by all combinations of $\hat{\theta}$ and $\hat{\phi}$ returned from all the feature points f.

However, calculating all the re-projections of the matched points, with all possible $\hat{\theta},\hat{\phi}$ trees regression of all feature points f can be computationally heavy. In order to make this process faster for real-time applications such as in AR, we also propose here a hybrid approach between our pose regression forest with RANSAC method [Morel and Yu 2009], the Random Forest Random Sample Consensus (RF-RANSAC).

In our RF-RANSAC approach, we combine the random sampling idea inside the decision forest. Thus, not only randomly sampling the observation nor features to be used, as a conventional approach, but also randomly sampling the trees within all the decision trees of all feature points.

As in [Morel and Yu 2009], we first sample a subset $S1$ as a percentage p_s of all matched feature points within $\mathbf{p}_m^I$ to use in the reprojection error calculation. Then, we randomly choose a matched feature point p_i^f that has a decision forest attached in.

We calculate the possible transformation T_{obj} by extracting the regression feature values $\mathbf{v}_i$ of p_i^f in the new image. We also randomly choose one θ regression tree $\tau_{(\theta,i)}^j$, and a $\tau_{(\phi,i)}^k$ for ϕ, within in p_i^f regression forest, in order to obtain $\hat{\theta}$ and $\hat{\phi}$ by $\mathbf{v}$. After this, we calculate the object transformation $\mathbf{T}_{obj}$ as in 3.3.

The reprojection errors of the observed point and $\mathbf{T}_{obj}$ is calculated similar to (12), using $S1$. If the number of point that has ϵ^I smaller than distance threshold d_{th}, is greater than acceptance percentage p_{th} of the subset, we accept $\mathbf{T}_{obj}$ as the object pose. Elsewhere, we keep re-estimating $\mathbf{T}_{obj}$ by using other random sample of feature point, θ-tree and ϕ-tree, while (**??**) is not satisfied. Finally, if we not found a $\mathbf{T}_{obj}$ that satisfies (**??**) within n_{trial}s; we then, set the object transformation $\mathbf{T}_{obj}$ as the transformation that had minimal error in the n_{trials}, as in (14). We illustrates the entire RF-RANSAC process at the algorithm 1.

Algorithm 1: RF-RANSAC algorithm

Data: P^f; set of $\tau_{(\theta,f)}$ and $\tau_{(\phi,f)}$; P_m^I; d_{th}; p_{th}; p_s
Result: Full transformation matrix T_{obj} of the rigid object
while *($found == false$) & ($n < n_{trials}$)* **do**
  Select random p_i^f within P_m^I;
  Select random $\tau_{(\theta,i)}^j$ and $\tau_{(\phi,i)}^k$;
  Regress $\hat{\theta}^j$ and $\hat{\phi}^k$;
  Select random second point $\mathbf{p}_2^I$ and its original $\mathbf{p}_2^o$;
  Calculate T_r, as in (1), and T_{obj}, as in (8);
  Select a random set of points $S1$, by p_s;
  forall the *points within* $S1$ **do**
    Calculate ϵ^I
  end
  if $\#[\epsilon^I < d_{th}] < p_{th} \cdot p_s \cdot N$ **then**
    $found = true$;
  else
    if $\sum(\epsilon^I) < \epsilon_{min}$ **then**
      Storage current transformation T_{obj} as minimal error transformation;
      Update $\epsilon_{min} = \sum(\epsilon^I)$;
    end
  end
end
if $fount \neq true$ **then**
  Use minimal error transformation as final $\mathbf{T}_{obj}$;
end
Return found T_{obj};

4 Results

We show case our RA results in figures 9 and 1, calculated by our RegressAR pose estimation approach. Results showed in these figures were obtained by using SURF feature [Bay et al. 2008] as features points, with 20 trees of τ_θ and τ_ϕ each, of 20 different feature points. Other than these forest attached feature points, we also use 280 feature points to help in the reproject error calculation and as second point for the rotation and translation calculation. In total, 300 feature points are used for the matching. The observation distance threshold d_{th} was set to 7 pix, $S1$ subset percentage p_s to be 60% of entire observation, acceptance percentage p_{th} as 80%, and n_{trials} to be 100.

Moreover, we set as $n_{pix} = 10$ at the feature point extraction size and $l_{ft} = 20$; thus we have a training vector $\mathbf{v}$ length of $\mathbf{v} \in \mathbb{R}^{400}$. For training, we generated around 2000 images samples with different and equally spaced θ and ϕ values, with no sigma and position variance, and more 5000 images with totally random poses. 10 different gaussian noise values are added to all of the images, giving us around $70,000$ samples used in the training. One important detail is that different cameras with different projection matrix K_{cam}, where used in the first and second row of the figure 9. Thus, showing that our approach demonstrates certain flexibility across different cameras, even strongly based on image appearances.

4.1 Method Evaluation

In this section we show evaluations of our method with different setups, in different cases. In order to process this evaluation we generated 1000 images with random poses, as in the training part, however, also with realistic backgrounds, photos taken from internet of real environments, in order to simulate a realistic capture. As in Lepetit et al.[2009] work, we compute the pose estimation error

Figure 9: *Augmented reality results of planar objects obtained by RF-RANSAC. Red line grids represents the entire object pose estimation. First and second rows results are obtained with different cameras with different projection matrix values.*

based on ground truth data to be $\mathbf{E}_{rot}(\%) = \|\mathbf{q}_{true} - \mathbf{q}\|/\|\mathbf{q}\|$ for rotation error, where $\mathbf{q}_{true}$ and $\mathbf{q}$ are quaternion vectors converted by the ground truth and estimated rotation matrices. Similarly, translation is defined to be $\mathbf{E}_{trans}(\%) = \|\mathbf{t}_{true} - \mathbf{t}\|/\|\mathbf{t}\|$, where $\mathbf{t}_{true}$ and $\mathbf{t}$ are the ground truth and estimation translation.

In figure 10, we show the box plot [1] of the errors obtained by our RF-RANSAC, and also the result of the pose estimated using the full analysis of errors calculated by equation (14). Also, in order to evaluate our results, we compared our results with the no iterative EPnP method proposed by Lepetit et al. [2009] and EPnP with RANSAC. Here, we used the same settings of our method as in the results obtained in figure 9, with 300 feature points for matching in both EPnP and EPnP-RANSAC methods. Additionally, we also used the same RANSAC settings as our RF-RANSAC in EPnP RANSAC.

Additionally, we generated other 100 images without the background, in order to evaluate the response of all the methods in cases with and without errors in feature points matches, presented also at figure 10. The errors are only computed if more than one forest attached feature and more than 3 feature points (including the forest attached feature points) are found in our methods; and for EPnP and EPnP RANSAC methods, only if the method returns that the pose are found.

From this results we can observe that our method has better rotation error than conventional and RANSAC EPnP methods in these cases, and comparable results in translation error. Also, we can observe that our method of fully calculating the error of all possible combinations, although slow, generates better accuracy and precision compared to EPnP methods. Another characteristic that we want to point, is that our method has less deviation than conventional methods.

[1] boxplot representation consists of the median (the middle line in red), the IQR, which is in total the 50% of the probability density function (presented by the box), the 1.5*IQR above and bellow (the extending lines) and the observation fall (in red crosses). Also, the mean values are represented as the middle of the boxes

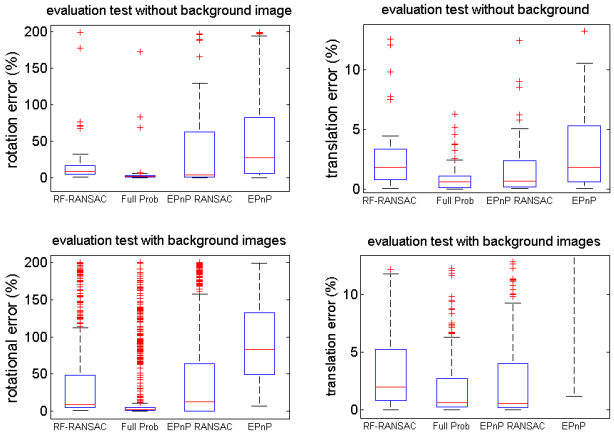
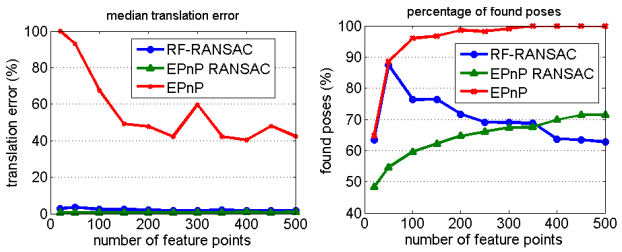


Figure 10: *Box plot of results evaluation of our approach, compared with EPnP and EPnP-RANSAC by Lepetit et al [2009]. First row: evaluation based on images without background images(flat black background); second row: with realistic images as background. First column: rotation error; second:translation errors.*

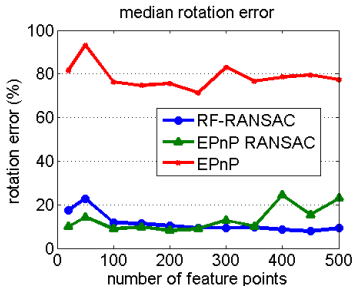
From the evaluation without background image (first row of figure 10), we can observe that our method has less accuracy when there are few feature point matching errors. However, this also means that our method is more robust to these kind of matching errors. This is due to the fact that our method, even using multiple points for verification, needs only 2 points in order to calculate the full pose, unlike solve P-n-P methods, that at least need 4 points in order to estimate the full pose.

This effect also can be observed in the case that we variate the total number of feature points to be matched. Showed in figure 11, we also tested our and EPnP methods with several different values of feature points for EPnP, and used the same number of feature points

for our method, but with fixed number of decision forest attached feature points as same as the last test, as well as using the same images with background.



(a) *translation error* **(b)** *percentage of found poses.*



(c) *rotation*

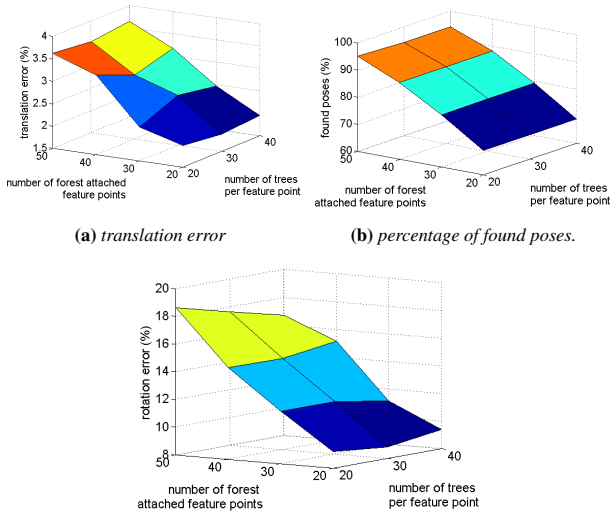
Figure 11: *Error values across different values of feature points. 11c shows the rotation errors; 11a, the translation error and 11b, the found poses across the evaluation images.*

In figure 11c we show the rotation errors. Note that our method has more rotation error in the beginning, however, that is because our method has higher found rate as shown in figure 11b. This happens once our method only can find the object pose, when at least one forest attached point is found; however, it also means that less accurate poses are also calculated in the error.

Moreover, once we have fixed the number of feature points with forest attached, our found percentage decreases with the number of feature points increase, once matching errors are corrected by the added points, decreasing our error too. However, around 250 to 400 feature points, where the found rate of our and EPnP-RANSAC are similar, we can observe that our accuracy for rotation is better than the conventional EPnP. Finally, from figures 11a we can observe that our method perform as good as EPnP RANSAC approach for translation estimation.

Additionally, in order to evaluate the effect of the number of trees, which are related to the estimation accuracy, and number of forest attached feature points, which also is related to the accuracy and founding percentage, we proceed a evaluation test across different tree and forest attached feature point numbers. Results obtained by this test are shown at figure 12.

From figure 12b, is possible to observe that with the increase of forest attached feature points, the probability of founding the object pose increases. However, also, the rotation and translation error has increased, as shown in figure 12c, as presented before. From figures 12c and 12a, it is possible to observe that the increase of trees in the forest does not affect the error or the finding probability. This is due the high probability of regression forest to have some trees with an accurate output, thus, not needing to have lot of trees in the forest to obtain a blind mean value; instead, the verification of the re-projected point of our approach performs better than conventional decision forest.



(a) *translation error* **(b)** *percentage of found poses.*

(c) *rotation error*

Figure 12: *Evaluation of translation and rotation errors, and number of founds across all the test set; across different forest trees numbers and different number of forest attached feature points.*

4.2 Results Extensions

Although we focused on rigid planar objects in this work, our approach could be extended for 3D rigid objects. This could be achieved by using reconstructed 3D models [Lepetit and Fua 2006] of the object in the training part, or using a unfolding function, which transforms the object surface texture to planar object, as presented in UV unwrapping functions [Lévy et al. 2002] or unfolding 3D models works [Mitani and Suzuki 2004]. Then, using the inverse, folding function, at equation (1) for θ and ϕ transformations, re-transforming it the original object space. We explore this second approach here, in order to evaluate the possible application of our approach for 3D objects. We show the results obtained in figure 13.

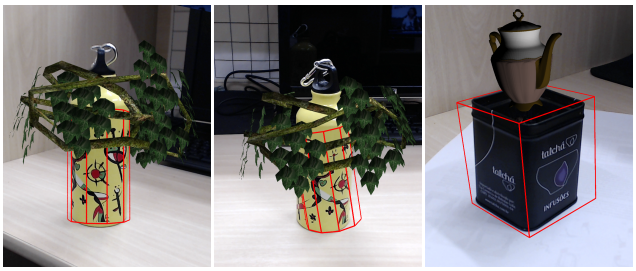


Figure 13: *Approximated pose estimation of 3D rigid objects, obtained by extending our approach for 3D rigid objects.*

More over, by finding the local rotation of each feature points, its information could be used in order to pose estimate deformable objects, that are locally rigid as in [Tokunaga et al. 2015]. The local rotation results are shown in figure 14. In this figure, the local rotations are calculated by conventional decision forest approach, equation (9), of the regressed values from all the trees of all the found feature points.

Despite further improvement needs to be done to increase the accuracy and precision, we observe from 13 and 14, that is plausible to extends our method for 3D obejcts or deformable objects. Note also that this also suggests that our locally planar assumption can

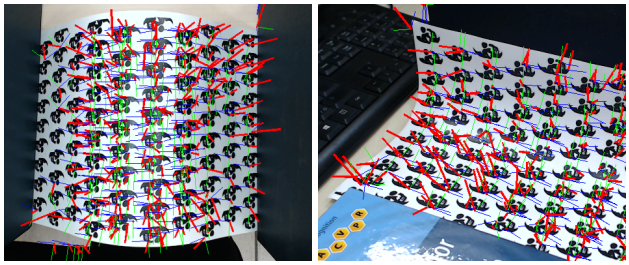


Figure 14: *Approximated local rotation estimation of deformable objects, by conventional decision forest. Each local rotation is represented by the coordinate system rendered over its 2D position. Red lines represent the Y(normal)-axis; green, X-axis; and blue, Z-axis.*

be relaxed for different cases of pose regression.

5 Discussion and Limitations

Our RegressAR approach, based on learning the perspective changes, for pose estimation is significantly different from conventional methods, that are based on the point correlations. Moreover, unlike conventional solve P-n-P approaches of pose estimation, that simplify all information of feature point to a single 3D point; we explore information given by the feature points, and its near surrounding surface appearance.

However, at the same time, it has a number of limitations and in open issues due to its novel nature. First, a limitation directly related to decision forest, is the memory consumption that our method has. In order to process the regression, we need to generate and store in memory several decision trees, which often can have more then $10,000$ nodes.

A second limitation is related to appearance distortions of the feature points. Once we train the feature point appearance captured in images, any distortion of the feature point can lead to wrong poses. Even, distortions in the training images can deeply affect our approach, such as lens distortions or scale distortions in XY-coordinates. Therefore, distortion corrections methods [Zhang 2000] need to be applied in both training and captured images. Similar to limitations of image distortions, positioning errors, orientation error and miss matches of feature points also can affect our results.

On the other hand, our approach shows to be more robust than conventional methods through feature points miss matches, because of the injection of prior information of observed points in the RF-RANSAC method. Additionally, our assumption of locally planar assumption is shown that can be relaxed, as shown at section 4.2, this gives flexibility to our method for future works.

Furthermore, decision forests can exploit parallelism for the evaluation, such as shown by Shotton et al.[Shotton et al. 2013], improving its regression speed. This can be applied in our method, speeding up the pose estimation itself, or our full probability estimation proposed in equation (13), which generate more accurate results for the cost of computational performance.

6 Conclusion

In this paper we presented RegressAR, a novel pose estimation approach based on machine learning for augmented reality. This approach is based on learning the local appearance around the feature points across different perspective changes, to proceed a regression of the planar object pose. In order to improve our results, we also introduce RF-RANSAC method, a hybrid method between RANSAC and decision forest. Here, we show that our method can perform better than conventional methods in several planar cases. Moreover, it can be extended to estimate pose of 3D objects.

Our method opens up several new research possibilities in several directions as future works. One future work is the exploration of parallelism with our method in order to speed up the entire process. Increasing the accuracy of per forests attached feature point or per single regression tree is another open issue. This could be done by applying non-euclidean space probability methods, such as spherical probability distributions or von Mises distribution [Abramowitz 1974], for θ and specially ϕ regression. Also, exploring other feature values for the training could improve the accuracy of each tree, such as explored in [Criminisi and Shotton 2013] or [Shotton et al. 2013].

This improvement in the per feature point regression accuracy, can lead to poses estimations of other types of object, by estimating the local transformation of each feature point with equation (1). This can be done, once our approach can calculate the rotation of the object entirely by local feature information. These rotations matrices could be used to estimate even deformable objects when combined with methods, such as proposed by [Rendl et al. 2014], or enable object rotation estimation by single feature point, as in [Tokunaga et al. 2015].

Finally, our RegressAR approach of pose estimation based on machine learning presents a novel alternative approach for pose estimation for augmented reality systems. This approach is based on a simple yet novel idea of learning perspective changes of the object in different poses, replacing the complex non-linear systems solving problem to a learning regression problem. We show that our results has comparable accuracy to state-of-art solve P-n-P methods, and some times better than these methods.

References

ABRAMOWITZ, M. 1974. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*. Dover Publications, Incorporated.

BAU, O., POUPYREV, I., LE GOC, M., GALLIOT, L., AND GLISSON, M. 2012. Revel: Tactile feedback technology for augmented reality. In *ACM SIGGRAPH 2012 Emerging Technologies*, ACM, New York, NY, USA, SIGGRAPH '12, 17:1–17:1.

BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. 2008. Speeded-up robust features (surf). *Comput. Vis. Image Underst. 110*, 3 (June), 346–359.

BENKO, H., WILSON, A. D., AND ZANNIER, F. 2014. Dyadic projected spatial augmented reality. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, NY, USA, UIST '14, 645–655.

BRACHMANN, E., KRULL, A., MICHEL, F., GUMHOLD, S., SHOTTON, J., AND ROTHER, C. 2014. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision, ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8690 of *Lecture Notes in Computer Science*. Springer International Publishing, 536–551.

COMPORT, A., MARCHAND, E., PRESSIGOUT, M., AND CHAUMETTE, F. 2006. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *Visualization and Computer Graphics, IEEE Transactions on 12*, 4 (July), 615–628.

CRIMINISI, A., AND SHOTTON, J. 2013. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Publishing Company, Incorporated.

FENZI, M., LEAL-TAIXE, L., ROSENHAHN, B., AND OSTERMANN, J. 2013. Class generative models based on feature regression for pose estimation of object categories. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 755–762.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June), 381–395.

GU, C., AND REN, X. 2010. Discriminative mixture-of-templates for viewpoint classification. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, Springer-Verlag, Berlin, Heidelberg, ECCV'10, 408–421.

HARA, K., AND CHELLAPPA, R. 2014. Growing regression forests by classification: Applications to object pose estimation. In *Computer Vision, ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8690 of *Lecture Notes in Computer Science*. Springer International Publishing, 552–567.

HE, K., SIGAL, L., AND SCLAROFF, S. 2014. Parameterizing object detectors in the continuous pose space. In *Computer Vision, ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8692 of *Lecture Notes in Computer Science*. Springer International Publishing, 450–465.

HEUN, V., KASAHARA, S., AND MAES, P. 2013. Smarter objects: Using ar technology to program physical objects and their interactions. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI EA '13, 961–966.

KUKELOVA, Z., BUJNAK, M., AND PAJDLA, T. 2008. Automatic generator of minimal problem solvers. In *Computer Vision ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds., vol. 5304 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 302–315.

LEITHINGER, D., FOLLMER, S., OLWAL, A., AND ISHII, H. 2014. Physical telepresence: Shape capture and display for embodied, computer-mediated remote collaboration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, NY, USA, UIST '14, 461–470.

LEPETIT, V., AND FUA, P. 2006. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 2006.

LEPETIT, V., LAGGER, P., AND FUA, P. 2005. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 775–781 vol. 2.

LEPETIT, V., MORENO-NOGUER, F., AND FUA, P. 2009. Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision 81*, 2 (Feb.), 155–166.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (July), 362–371.

LI, S., XU, C., AND XIE, M. 2012. A robust o(n) solution to the perspective-n-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 7 (July), 1444–1450.

LOWE, D. 1991. Fitting parameterized three-dimensional models to images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 13*, 5 (May), 441–450.

LOWE, D. 1999. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1150–1157 vol.2.

LU, C.-P., HAGER, G., AND MJOLSNESS, E. 2000. Fast and globally convergent pose estimation from video images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 6 (Jun), 610–622.

MITANI, J., AND SUZUKI, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph. 23*, 3 (Aug.), 259–263.

MOREL, J.-M., AND YU, G. 2009. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci. 2*, 2 (Apr.), 438–469.

MURASE, H., AND NAYAR, S. K. 1993. Parametric eigenspace representation for visual learning and recognition. vol. 2031, 378–391.

PEPIK, B., GEHLER, P., STARK, M., AND SCHIELE, B. 2012. 3d2pm 3d deformable part models. In *Computer Vision, ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., vol. 7577 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 356–370.

RENDL, C., KIM, D., FANELLO, S., PARZER, P., RHEMANN, C., TAYLOR, J., ZIRKL, M., SCHEIPL, G., ROTHLÄNDER, T., HALLER, M., AND IZADI, S. 2014. Flexsense: A transparent self-sensing deformable surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, NY, USA, UIST '14, 129–138.

SHIMIZU, S., KOYASU, H., KOBAYASHI, Y., AND KUNO, Y. 2015. Object pose estimation using category information from a single image. In *Frontiers of Computer Vision (FCV), 2015 21st Korea-Japan Joint Workshop on*, 1–4.

SHOTTON, J., GLOCKER, B., ZACH, C., IZADI, S., CRIMINISI, A., AND FITZGIBBON, A. 2013. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, CVPR '13, 2930–2937.

TEJANI, A., TANG, D., KOUSKOURIDAS, R., AND KIM, T.-K. 2014. Latent-class hough forests for 3d object detection and pose estimation. In *Computer Vision, ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8694 of *Lecture Notes in Computer Science*. Springer International Publishing, 462–477.

TOKUNAGA, D. M., NAKAMURA, R., JR., J. L. B., RANZINI, E., AND TORI, R. 2015. Local 3d pose estimation of feature points based on RGB-D information for object based augmented reality. In *Virtual, Augmented and Mixed Reality - 7th International Conference, VAMR 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings*, 130–141.

TORKI, M., AND ELGAMMAL, A. 2011. Regression from local features for viewpoint and pose estimation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2603–2610.

VAN KREVELEN, D. W. F., AND POELMAN, R. 2010. A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality 9*, 2 (June), 1–20.

WU, C., CLIPP, B., LI, X., FRAHM, J.-M., AND POLLEFEYS, M. 2008. 3d model matching with viewpoint-invariant patches (vip). In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 11 (Nov.), 1330–1334.

ZHEN, X., WANG, Z., YU, M., AND LI, S. 2015. Supervised descriptor learning for multi-output regression. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 1211–1218.