2014 IEEE International Conference on Robotics and Automation
Hong Kong Convention and Exhibition Centre
May 31-June 7, 2014, Hong Kong, China

Estimating finger grip force from an image of the hand using Convolutional Neural Networks and Gaussian Processes

Nutan Chen, Sebastian Urban, Christian Osendorfer, Justin Bayer, and Patrick van der Smagt

***Abstract*— Estimating human fingertip forces is required to understand force distribution in grasping and manipulation. Human grasping behavior can then be used to develop force- and impedance-based grasping and manipulation strategies for robotic hands. However, estimating human grip force naturally is only possible with instrumented objects or unnatural gloves, thus greatly limiting the type of objects used.**

In this paper we describe an approach which uses images of the human fingertip to reconstruct grip force and torque at the finger. Our approach does not use finger-mounted equipment, but instead a steady camera observing the fingers of the hand from a distance. This allows for finger force estimation without *any* physical interference with the hand or object itself, and is therefore universally applicable.

We construct a 3-dimensional finger model from 2D images. Convolutional Neural Networks (CNN) are used to predict the 2D image to a 3D model transformation matrix. Two methods of CNN are designed for separate and combined outputs of orientation and position. After learning, our system shows an alignment accuracy over 98% on unknown data.

In the final step, a Gaussian process estimates finger force and torque from the aligned images based on color changes and deformations of the nail and its surrounding skin. Experimental results shows that the accuracy achieves about 95% in the force estimation and 90% in the torque.

I. Introduction

To fully exploit grip stability, manipulation capabilities, and grip force and stiffness of dexterous robotic hands, the human hand often serves as an example. Detailed studies of finger positions during grasping (e.g., [1]–[4]) give key information of the *position* of the fingers during grasp, but not on the forces and torques exerted. Studies with instrumented objects (e.g., [5], [6]) can give some information, but only limited: in many cases, simple force-sensitive resistors are used which can only estimate surface normal forces, or a very limited number (1 or 2) of load cells can estimate full force and torque but at only one predefined location.

We use a different approach to estimate grip force and torque. For our method, we exploit the fact that finger tip depression causes nail (bed) color change related to that pressure. Through this, full estimation of grip and torque is possible at any interaction point.

Our method does not require mounting a sensor at the finger or the object; instead, we localize the finger in an image and estimate from there. A crucial aspect therefore is image alignment. Prior methods of fingernail image registration are 2D-to-3D registration with a grid pattern and fiducial markers onto the finger [7], rigid body transformation including the Harris feature point based method [8], Canny edge detection [9], template matching using markers [10], non-rigid registration fitting a finger model [8], and Active Appearance Models (AAM) [11]. Other methods use sensors mounted on the finger [10], [12] or require restrictions such as a bracket to support the hand [7] or the finger [9], [11].

The authors are with the Faculty for Informatics, Technical University Munich, 80333 Germany nutan(at)in.tum.de, surban(at)tum.de, osendorf(at)in.tum.de, bayer.justin(at)gmail.com. PvdS is also with fortiss.

Fig. 1: Setup: a fixed camera observes the fingers to estimate finger grip force and torque.

We argue that, for human grasping, a more robust and generally applicable system is required, which does not obstruct movement or interferes with experiments otherwise. To address this challenge, we learn a 3D model of a finger and match 2D images using Convolutional Neural Networks (CNNs) to estimate grip force. The images can be aligned robustly without distortion as non-rigid registration, even when they are partially blocked. 2D–3D alignment was also used in [7] but needs special markers on the finger and a laser grid pattern. In contrast, our method just needs a designed marker pasted on the finger nail without calibration, since the marker is designed to add features for the finger but no position requirement. Our setup is depicted in Fig. 1. The calibration method for a single finger is shown in Fig. 2.

Following preprocessing, various methods have been developed to estimate the force. Model-based methods [11] contain linearized sigmoid models, EigenNail models [13], and linearized sigmoid models. In [10] we previously estimated force and torque using Gaussian processes (GP) and neural networks. Following the high accuracy obtained there, here we use Gaussian processes to estimate force from the aligned images.

II. Methodology

In this section we describe the data acquisition setup and the methods used to extract the nail from the video data and estimate the forces.

A. Hardware Setup

The recording system is shown in Fig. 3. A stationary IMAGINGSOURCE camera captures video data at 15 fps with a resolution of 1024×768 pixels. Additionally, an

Fig. 2: Calibration setup. A force-torque sensor (FTS) measures the real forces and torques exerted by the finger, while being observed by the camera.

ATI Nano-17 force/torque sensor records force/torque data from a finger pad at 100 Hz. Green LEDs are used to synchronize the timing of the visual and force/torque data through blinking at the beginning and end of a recording sequence. The experiments take place with a mini lighting studio ($40 \times 40 \times 40\,\mathrm{cm}^3$) with diffuse white lamp light.

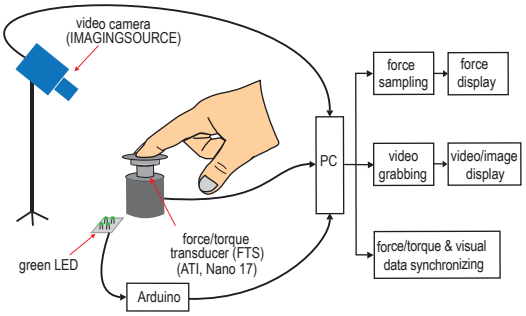

Fig. 3: Recording system. The stationary camera records visual data of the finger which applies pressure on the force-torque sensor.

B. Image alignment using convolutional neural networks

In the video stream the finger position and orientation vary with time. To reduce the effect of these variations we use a convolutional neural network (CNN) to predict and correct for the finger position and orientation before estimating forces and torques.

1) 3D model construction: To train the CNN we need a 3D model of the finger. Thus, before the actual experiments take place, we construct a triangular, textured mesh 3D model from between 12 and 15 images of a finger using the commercially available Agisoft Photoscan 0.9.1 software. This process lasts a few tens of seconds.

2) Finger detection and tracking: Since the video stream contains not only the finger but also an arbitrary background we first need to extract the finger from the image. We employ tresholding in the YCbCr color space, which is a nonlinear RGB signal often used for skin detection, to segment the finger. We then use the mean shift algorithm [14] to track the finger in the video stream.

But the "raw" 2D image of a finger cannot, when the finger is tilted, be perfectly mapped on the original image. Using the image of a tilted finger would lead to considerable data loss. To resolve this problem, we transform tilted images back to the original, as follows.

3) Transformation matrix estimation using convolutional neural networks: Let the position of the finger in world coordinates be given by the tuple x, y, z and the orientation by the angles α, β and γ, which refer to the yaw, pitch, and roll, respectively. We set $x = y = z = \alpha = \beta = \gamma = 0$ to be the front view of the 3D finger model. The world coordinate $\mathbf{p_1}$ of a point $\mathbf{p_0}$ on the finger surface is given by $\mathbf{p_1} = V\mathbf{p_0}$ with the affine transformation matrix V given by

$$V = \begin{bmatrix} R & \mathbf{q} \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where

$$R = \mathrm{Rot}_z(\alpha) \cdot \mathrm{Rot}_y(\beta) \cdot \mathrm{Rot}_x(\gamma), \tag{2}$$

$$\mathbf{q} = (x\ y\ z)^T, \tag{3}$$

and $\mathrm{Rot}_w(\theta)$ denotes the rotation matrix for a rotation about the w-axis by angle θ.

To simplify the alignment task and reduce environment ambiguities such as lighting and reflection we place a marker (see Fig. 4) on each nail. The method is not sensitive to the placement of the marker, as long as the marker is located at the same place of the fingernail for the training and testing data. The design of the marker is chosen so that the four lines identify the orientation of the nail, while the x and y coordinates of the nail can be estimated by locating the red point in the 2D image. The black line on the left unambiguously distinguishes the lines from each other.

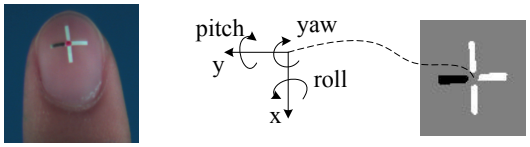

Fig. 4: Marker. The left picture shows the design of the marker. The right picture shows the marker image from a finger image. The middle image shows the marker coordinate.

We render about 40,000 two-dimensional training images from the 3D model created in step 1 with varying values for the finger position z and orientation α, β, γ. z is in a range of $[-2, 3]$ with a step of 0.5 virtual distance in the model coordinate, while $\alpha \in [-25, 30]$, $\beta \in [-15, 35]$ and $\gamma \in [-23, 37]$ with a step of 3.5 degrees each. The virtual distance is the distance in the z direction with respect to the 3D model. The unit of the virtual distance depends on the size ratio from the human finger to the 3D finger model. The points making up the marker in the rendered image are extracted and resized to 81×88 pixels. For testing we use real finger images captured with the camera.

A convolutional neural network [15] is a neural network architecture for regression and classification that is relatively

robust to shifts, scales and distortions of the input data and can be trained efficiently on large data sets. Therefore, CNNs can detect the transformation parameters between a mis-aligned image and a reference image. We compare two approaches: one with separate orientation and position outputs through two CNNs, and one combining outputs through one CNN.

Fig. 5 illustrates the architecture of the proposed network for the combined output method. It contains six layers: a first convolutional layer followed by a first max-pooling layer, another convolutional layer followed by a second max-pooling layer, and two fully connected layers. In our experiments, both convolutional layers have 5×5 sized filters. The first convolutional layer employs 8 kernels, while the second makes use of 25 kernels.

We now describe convolutional neural networks more formally. Typically, a CNN is designed as subsequent stages of convolution and max-pooling. The top layers are usually ordinary multi-layer perceptrons.

The convolution of a 2D image for a feature map h is

$$h(m,n) = \sum_{u=0}^{l_x} \sum_{v=0}^{l_y} w(u,v)\, g(u+m, v+n) + b, \quad (4)$$

where g is the input map, w the kernel weights, and b the bias. (l_x, l_y) is the size of the filter. (m,n) is the pixel position on the feature map.

The max-pooling activation can be computed as

$$p(m,n) = \max_{i=1}^{r_1}\Big(\max_{j=1}^{r_2} h(r_1 m + i, r_2 n + j)\Big), \quad (5)$$

where (r_1, r_2) is the pooling size and p is the feature map in the max-pooling layer.

Max-pooling, a non-linear down-sampling method, decreases the computational complexity. These layers take the output of convolutional layers as input, and reduce the resolution of the input. In our case, that is a reduction from 77×84 to 38×42 and from 34×38 to 17×19.

The fully-connected MLP contains 50 hidden units. The last layer is linear and has 7 outputs $O_1, O_2, \ldots, O_7$. We identify the first of those outputs as $z = O_1$ while the remaining 6 contribute to the three orientations. The orientations $\theta \in \{\alpha, \beta, \gamma\}$ are calculated as follows:

$$\theta = \arccos\left(\frac{O_i}{\sqrt{O_i^2 + O_{i+1}^2}}\right) - \frac{\pi}{2}, \quad (6)$$

where $i \in \{2, 4, 6\}$. Note that since this form of encoding of angles is differentiable, we can use the chain rule to backpropagate error gradients back into the network.

Assuming that the output[1] y and input $\mathbf{x}$ are related linearly, i.e., $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon$ where $\mathbf{w}$ refers to the weight vector and the residual error ϵ is the difference between predictions and true values.

[1]In the sequel we assume the output y to be one-dimensional for notational simplicity, but the general equations generalize.

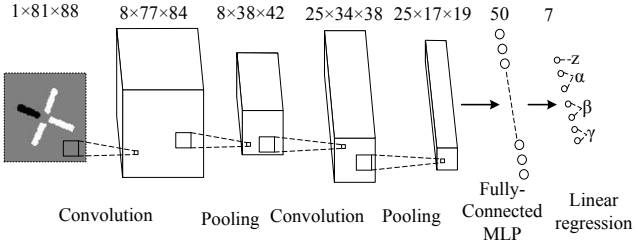

Fig. 5: Architecture of the proposed CNNs with combined orientation and position output.

With a conditional probability density, the linear regression model [16] is denoted by

$$p(y \mid \mathbf{x}, \xi) = N\big(y \mid \mu(\mathbf{x}), \sigma^2\big), \quad (7)$$

where N is the normal distribution, $\xi = (\mathbf{w}, \sigma^2)$ are the parameters, and σ^2 is the variance. The expected output is $\mu(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. In our model, $\mathbf{x}$ is the output of the MLP and the input of linear regression, and $y := (O_1, O_2, \ldots, O_7)^T$ is the output of linear regression.

The training data is assumed as independent and identically distributed (iid). To determine optimal values for the weights $\mathbf{w}$, we minimize the negative log likelihood (NLL)

$$\begin{aligned} \mathrm{NLL}(\xi) &\triangleq -\sum_{i=1}^{N} \log p(y_i \mid x_i, \xi) \\ &= \frac{-1}{2\sigma^2}\mathrm{SSE}(\mathbf{w}) - \frac{N}{2}\log(2\pi\sigma^2), \end{aligned} \quad (8)$$

where the sum of squared errors (SSE) is defined by

$$\mathrm{SSE}(w) \triangleq \sum_{i=1}^{N} (y_i - \mathbf{w}^T\mathbf{x}_i)^2, \quad (9)$$

with N the number of data points that we optimize on.

Since the orientations are periodic, we investigated using a von Mises distribution [17] to calculate the log likelihood function. However, experimental evidence showed that using a straightforward $L2$ norm leads to better results. Therefore the cost function f is denoted by

$$f(\theta) = \sum_{n=1}^{N} (\theta_n - \theta_0)^2. \quad (10)$$

Through minimizing the cost function of α, β, and γ and the negative log likelihood of z, we can update the weights of the CNN.

Observation from the training process of the combined output CNNs model, the position and orientation variables converge separately at the beginning. More specifically, z convergence occurs almost after the α, β and γ are stable; therefore, with the assumption that z is independent of α, β and γ, we design CNNs as Fig. 6 for the comparison of combined and separate training. In contrast to the combined method, this method trains position and orientation separately with the only connection that the trained output of orientation is set as the bias of the linear regression layer

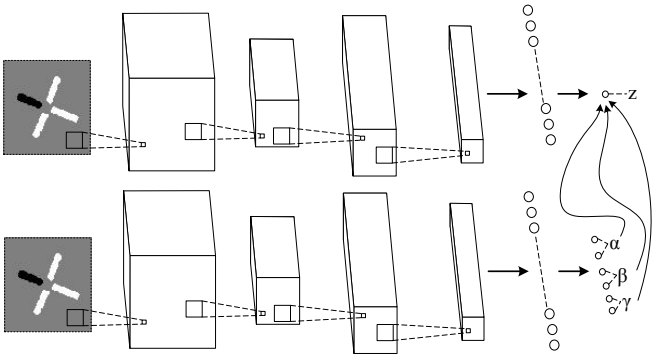

Fig. 6: CNNs of separate orientation and position output. The prediction of the orientation is the bias of of last layer for position.

of the position CNNs model. Thus, the expected outputs of z is given by

$$\mu(x) = w_\alpha \alpha + w_\beta \beta + w_\gamma \gamma + w_1 x_1 + \cdots + w_n x_n.$$

4) Texture Mapping: Given the estimated transformation matrix from the CNN, the image can be aligned using texture mapping [18]. This is an efficient method [19] to create the appearance from a source image without the tedious processes such as modeling or rendering a 3D surface for every detail. It allows "glueing" the source 2D frame onto the 3D finger surface in the estimated position and orientation. The mapped 3D finger model is then drawn to the destination image through the perspective projection in the reference position and orientation.

The source image is in a texture space labeled by (u, v); the 3D model is in an object space labeled by (x_w, y_w, z_w), and the aligned image is in a screen space labeled by (x_s, y_s). Fig. 7 shows the texture mapping process and the spaces.

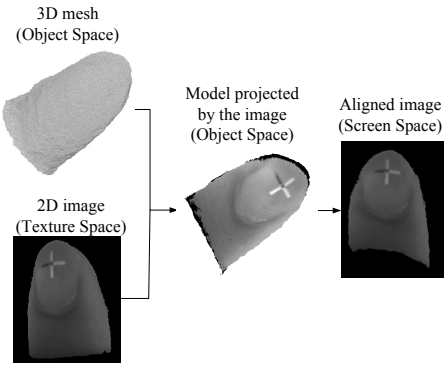

Fig. 7: Texture mapping.

5) Nail and Skin Extraction: In the aligned images, the edges of the fingers may not be the same, because different images may have different occluded areas during the movement. However, the common visible areas have the same appearance in the aligned images which contain the pressure information in the form of colour changes. To reduce the noise induced by the finger edges and the environment the intersection of all visible areas over the whole video stream is extracted.

C. Image Estimates Force/Torque–Gaussian Process

The aligned images are divided into a training and test set with about 82% of the data is used for training. The testing samples are selected from time-continuous blocks of data.

A Gaussian Process (GP) [20] is a stochastic process given by its mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$,

$$m(\mathbf{x}) = \mathbf{E}[f(\mathbf{x})], \tag{11}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{E}\Big[\big(f(\mathbf{x}) - m(\mathbf{x})\big)\big(f(\mathbf{x}') - m(\mathbf{x}')\big)\Big]. \tag{12}$$

Assuming the GP has a zero mean function, the squared exponential covariance (SE) is derived as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\Big(-\frac{1}{2l^2}\|\mathbf{x} - \mathbf{x}'\|_2^2\Big). \tag{13}$$

The length-scale l and the signal variance σ_f^2 are the hyper parameters. Points that have distances to each other smaller than l can be considered to have similar values.

The inputs of training points are $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})$. $\mathbf{x_i}$ is an aligned image after being reshaped to a 1D vector. In addition, the estimated force and torque, $\mathbf{y} := (y_1, y_2, \ldots, y_N)^T$ is the corresponding target. Thus, the target value $f(\mathbf{x}^*)$ for $\mathbf{x}^*$ is distributed as

$$\mathbf{E}[f^*] = \mathbf{k}^{*T}(K + \sigma_n^2 I)^{-1}\mathbf{y}, \tag{14}$$

$$\mathrm{Var}[f^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^{*T}(K + \sigma_n^2 I)^{-1}\mathbf{k}^*, \tag{15}$$

where $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$, $k_i^* = k(\mathbf{x_i}, \mathbf{x}^*)$, I is the identity matrix and σ_n the noise variance hyper parameter.

We maximize the log likelihood function

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\big|K + \sigma_n^2 I\big| - \frac{n}{2}\log 2\pi, \tag{16}$$

and consequently obtain the optimal values for the three hyper parameters using the training set.

III. Experiments and Results

Experiments are carried out to evaluate the proposed approaches. There are 3 subjects denoted by S_1, S_2, and S_3 respectively. The subjects are trained separately using different models. The accuracy $\sqrt{R^2}$ for both CNNs and GP is given by

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - d_i)^2}{\sum_{i=1}^N (y_i - \overline{y})^2}, \tag{17}$$

where d_i is the target value, y_i the estimation value, and $\overline{y}$ the mean value of the test set.

A. Alignment Result

The training and validation data sets are generated from the 3D model as stated in the previous section. For one finger model, with different position and orientation, it generates about 40,000 images, 80% of which is randomly chosen to be the training set and the rest is validation set. The test set is made from real images captured by the camera.

Table I shows the validation results for CNNs alignment. Since the testing data has no labels, only validation data

TABLE I: Accuracy ($\sqrt{R^2}$) of combined and separate outputs of the CNN. t represents the training time of the implementation in Theano.

		z	α	β	γ	t/min
S_1	Combined	0.988	0.999	0.999	1.000	352
	Separate	0.973	0.999	0.999	1.000	323
S_2	Combined	0.986	0.998	0.999	1.000	360
	Separate	0.971	0.998	0.999	1.000	333
S_3	Combined	0.985	0.997	0.997	1.000	372
	Separate	0.981	0.996	0.996	1.000	339
avg	Combined	0.986	0.998	0.998	1.000	362
	Separate	0.975	0.998	0.998	1.000	332

is quantified in this process. The combined method and separate method achieve high accuracy and have almost the same accuracy for α, β, and γ, while 1.14% difference on z. It indicates that z is approximately independent of the orientation. Further results are explained in the next section.

In terms of the training time, training two separate CNNs is more effective, reducing training time from 6 hours to 5.5. Once the system is trained, the CNN runs in realtime, predicting one image after 4 ms on a GPU.

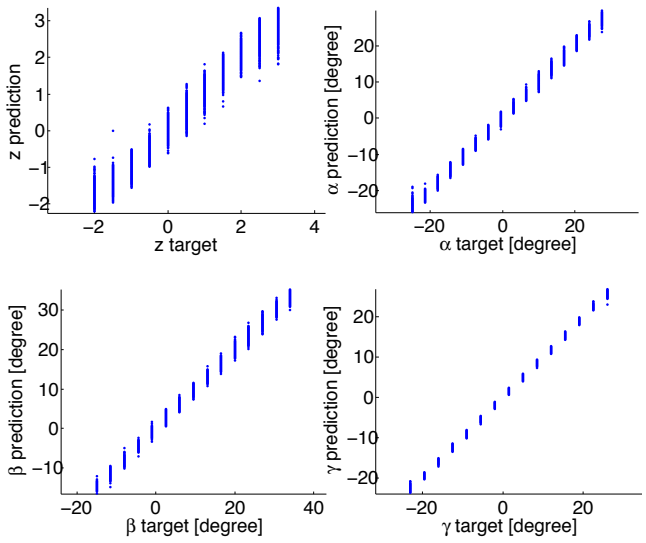

Fig. 8: Validation result of alignment for S_1 by Convolutional Neural Networks

As an example, some detailed results are shown in Fig. 8 for S_1. The other two subjects are similar. The prediction error is shown for z and for the three rotation angles. The reduced accuracy in z is related to the small range that z has. One can see that there are systematic errors—the model overestimates for low, and underestimates for high orientations/positions. We hypothesize that it can be tackled with either more data and/or more powerful models, e.g. more hidden layers and units.

Fig. 9 shows three different testing images before and after alignment. The aligned images have the same appearance in the interested areas, but different colors in the same pixel.

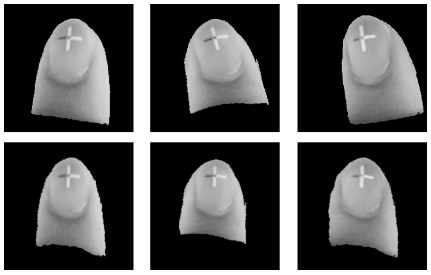

Fig. 9: Alignment result of a finger. The top row are the images before alignment, the bottom row pictures are the images after alignment.

TABLE II: Accuracy ($\sqrt{R^2}$) of combined and separate outputs of Force Estimation by Gaussian process

		Combined			Separate		
		x	y	z	x	y	z
S_1	f	0.936	0.929	0.956	0.939	0.929	0.957
	τ	0.927	0.941	0.886	0.914	0.943	0.888
S_2	f	0.933	0.913	0.970	0.930	0.915	0.966
	τ	0.914	0.917	0.785	0.913	0.925	0.815
S_3	f	0.942	0.946	0.947	0.950	0.952	0.944
	τ	0.766	0.934	0.873	0.751	0.942	0.886
Avg.	f	0.937	0.934	0.958	0.940	0.932	0.956
	τ	0.869	0.937	0.848	0.859	0.937	0.863

B. *Force/torque Estimation Result*

After obtaining the aligned images, the force and torque can be estimated through the GP method we used before [10]. Both of the training and testing inputs are from the aligned frames of the video camera. There are about 1500 frames for each subject.

Finger rotation was not restricted. The rotation ranges estimated by the CNN is up to about 30°, 25° and 20° for α, β and γ.

We evaluate the force (f) / torque (τ) effects for all 3 subjects in Table II. The negative z-coordinate with respect to the transducer is the downward movement in direction to the pressure transducer. The accuracy of f_z reaches about 95.8% with the range up to 10 N, while f_x and f_y achieve over 93%. Some results of the torque are not as accurate as the rest, which is caused by the recording data. The contact point of the finger and the sensor is not fixed, especially, to prove the robust alignment, the subjects rotate the finger in a relative large range without support for the arm or the finger; therefore, the estimation sifts according to the contact point. Fig. 10 exemplarily illustrates the results for S_1.

The accuracy of force estimation using the aligned images from two CNNs methods have no more than 1% difference. Thus, we can deduce that the orientation and position are independent for all practical reasons. Based on the effective training, the separate method is better than the combined method.

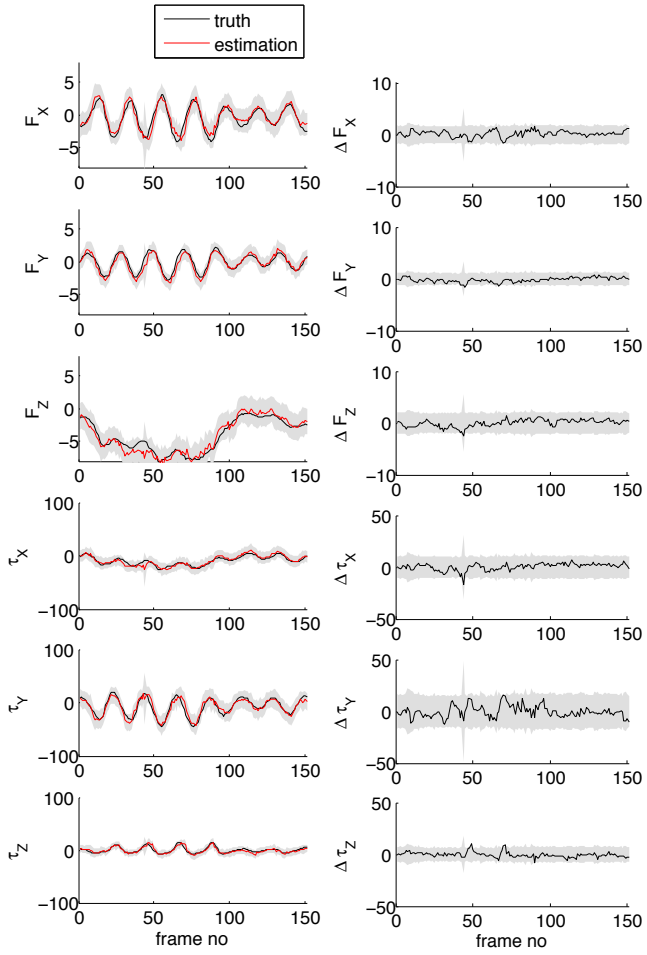

Fig. 10: Test result of force/torque estimation for S_1 by Gaussian process. The left pictures show the true value and estimation value of the force/torque. The right pictures show the estimation and true value difference. The shadow is the 95% confidence interval of the predictor.

IV. Conclusions

This paper has presented a new approach to align 2D finger images to a 3D model using machine learning approaches, in order to deduce accurate grip force from nail coloration with a steady camera. Our approach, based on Convolutional Neural Networks, predicts the rotation with an accuracy of 97.47% to 99.97%. We compare two approaches, by learning learning rotation and translation in one CNN or in two separate CNNs. Both approaches obtain similar results, except that the separated approach is computationally more efficient (and would allow for additional parallelization) during learning. Both methods need only 4 ms to evaluate one image during use.

The proposed alignment approach obtains as much as information of the nail and its surrounding skin without distortion as non-rigid registration. It does not require any special lighting conditions.

With the robust alignment system, the force and torque estimation of a finger is about 95% cq. 90%, allowing for unrestricted movement of the hand and a placement-free camera.

V. Acknowledgement

Part of this work has been supported in part by the TACMAN project, EC Grant agreement no. 610967, within the FP7 framework programme. This work has been supported in part by the Swedish Research Council, VR 2011-3128. The authors kindly thank Benoni Ben Edin and Göran Westling, Umeå University for their active support of this work.

References

[1] J. N. Ingram, K. P. Körding, I. S. Howard, and D. M. Wolpert, "The statistics of natural hand movements," *Experimental Brain Research*, vol. 188, no. 2, pp. 223–236, Mar. 2008.

[2] A. Gustus, G. Stillfried, J. Visser, H. Jörntell, and P. van der Smagt, "Human hand modelling: kinematics, dynamics, applications," *Biological cybernetics*, vol. 106, no. 11-12, pp. 741–755, Nov. 2012.

[3] N. Fligge, H. Urbanek, and P. van der Smagt, "Relation between object properties and emg during reaching to grasp," *Journal of Electromyography and Kinesiology*, vol. 23, no. 2, pp. 402–410, 2013.

[4] H. Höppner, J. McIntyre, and P. van der Smagt, "Task dependency of grip stiffness—a study of human grip force and grip stiffness dependency during two different tasks with same grip forces," *PLOS ONE*, vol. 8, no. 12, p. e80889, 2013.

[5] B. B. Edin, G. Westling, and R. S. Johansson, "Independent control of human finger-tip forces at individual digits during precision lifting," *The Journal of Physiology*, vol. 450, pp. 547–564, May 1992.

[6] J. R. de Gruijl, P. van der Smagt, and C. I. de Zeeuw, "Anticipatory grip force control using a cerebellar model," *Neuroscience*, 2009.

[7] Y. Sun, J. Hollerbach, and S. Mascaro, "Predicting fingertip forces by imaging coloration changes in the fingernail and surrounding skin," *IEEE Tr Biomed Eng*, vol. 55, no. 10, pp. 2363–2371, 2008.

[8] ——, "Estimation of fingertip force direction with computer vision," *IEEE Tr Robotics*, vol. 25, no. 6, pp. 1356–1369, 2009.

[9] T. Grieve, L. Lincoln, Y. Sun, J. Hollerbach, and S. Mascaro, "3d force prediction using fingernail imaging with automated calibration," in *IEEE Haptics Symposium*, 2010, pp. 113–120.

[10] S. Urban, J. Bayer, C. Osendorfer, G. Westling, B. B. Edin, and P. van der Smagt, "Computing grip force and torque from finger nail images using gaussian processes," in *IROS*, 2013.

[11] T. R. Grieve, J. M. Hollerbach, and S. A. Mascaro, "Force prediction by fingernail imaging using active appearance models," in *World Haptics Conference (WHC), 2013*, 2013, pp. 181–186.

[12] S. A. Mascaro and H. H. Asada, "Measurement of finger posture and three-axis fingertip touch force using fingernail sensors," *IEEE Tr Robotics and Automation*, vol. 20, no. 1, pp. 26–35, Jan. 2004.

[13] Y. Sun, J. Hollerbach, and S. Mascaro, "EigenNail for finger force direction recognition," in *ICRA*, 2007, pp. 3251–3256.

[14] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," 2000, pp. 142–149.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[16] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[18] T. Yu, H. Wang, N. Ahuja, and W.-C. Chen, "Sparse lumigraph relighting by illumination and reflectance estimation from multi-view images," in *ACM SIGGRAPH 2006 Sketches*, 2006.

[19] P. S. Heckbert, "Survey of texture mapping," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 56–67, Nov. 1986.

[20] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.