Automatic Camera-Screen Localization

Francisco Gómez-Fernández[1], Zicheng Liu[3], Alvaro Pardo[2], and Marta Mejail[1]

[1] Universidad de Buenos Aires
[2] Universidad Católica del Uruguay
[3] Microsoft Research

Abstract. Knowing the location of the TV screen with respect to a camera it is important for many applications. This work addresses this problem in a configuration where there are people looking at the TV and a RGB-D camera facing them, located near the TV screen. We propose a method to automatically estimate the screen location and camera rotation using only people's head pose obtained from a Face Tracking analysis on the RGB-D video. We validated these algorithms on a dataset with groundtruth and obtained very promising results.

Keywords: Human-Computer Interaction, Head pose estimation, Screen localization.

1 Introduction

In Human-computer interaction, it is critical to know whether a user is paying attention to the screen or not. This requires the knowledge of the screen position with respect to the camera. To estimate the screen position, one obvious solution is to ask the user to perform screen-camera calibration. Unfortunately, this solution is not practical for non-built-in cameras because people usually move them around and they do not want to perform calibration every time the camera is moved.

In this paper, we propose a solution that is completely automatic and does not require the involvement of the user. The main observation is that even though the camera cannot see the screen, the camera can see the people who are watching the screen.

The basic setting for this work is made of people looking at the TV in a standard house room, for example a living room. A RGB-D sensor, such as a Kinect, is placed in this scenario looking at people. The goal of this work is to automatically find the relative position of the TV screen with respect to the Kinect sensor, which could be moved or rotated.

We assume that the position of the TV is fixed and the relative position of the camera with respect to the TV is unknown. We also assume that people are steady and they are looking at the TV most of the time (people's view-directions pointing at the TV screen). Since the resolution of the RGB images is low we cannot rely on gaze estimation and we can just use head pose as a coarse indication of gaze[9]. Thus, throughout this work, we leverage the head pose orientation of each person in the scene to infer the screen position.

For the best of our knowledge, this problem was never studied before and it has several interesting applications. We can provide feedback to a user interacting with a gesture-controlled system to better place the camera or adapt the algorithms when the camera was dropped or moved far away from the system ideal situation. In video chat, if the camera is placed too far from the screen, the remote site will always see a non-frontal view of the person thus degrading the social experience. A video chat system would like to know where the camera is respect to the screen so that the system could instruct the user to move the camera to a better position.

Our tests with different camera-screen localizations show that we can estimate efficiently where the TV screen is with respect to the camera and predict the camera rotation.

To conclude this introductory section we are going to review the closest works from the literature. The head pose estimation from depth and/or RGB images obtained by the Kinect, or similar sensors, has been addressed by many works [8,2,5,4]. For a survey on different head pose estimation methods, see [9]. The work [6] addresses the problem of gaze estimation under unrestricted head motion from RGB-D images obtained with Kinect. The authors propose a method to estimate the gaze direction in 3D. They assume that the head is close to the RGB-D camera so eyes can be successfully acquired. In our case, we can not rely on a good definition of the eyes and therefore we can only use head pose as we will explain in next sections.

Section 2 describes how to estimate people's view direction from head pose, i.e. where people are looking at and section 3 presents our method of automatic camera-screen localization. In section 4 we present results and discussion about the validation of our method. Finally, in section 5, conclusions and future work are presented.

2 People's View-Direction Computation

In this section, our main goal is to estimate where a person is looking at using a commodity RGB-D camera, such as the Kinect. These cameras have poor RGB and Depth resolutions, therefore, we cannot properly capture the eyes of a person who is more than a meter away from the sensor. Thus, gaze estimation is not feasible in this scenario.

Therefore, we will assume that a person is looking in the direction of their nose and use head pose as a coarse indication of gaze[9].

We rely on a 3D head pose estimation algorithm [2] to obtain the rotation and translation of each person's head. Head pose is described with yaw, pitch and roll angles, also known as heading, elevation, and bank in flight dynamics. These values are represented in a coordinate system centered at the camera's optical center where z axis is pointing towards the user and y axis is pointing up. Thus, pitch, yaw, and roll angles are rotations about x, y, and z axes, respectively.

The normal vector $\mathbf{n}$ to the face is computed from α, β, γ angles (roll, yaw, and pitch, respectively) as: $\mathbf{n} = R_x(\gamma)\, R_y(\beta)\, R_z(\alpha)\ [0, 0, -1]^T$, where $R_x(\theta), R_y(\theta),$

$R_z(\theta)$ are rotation matrices which rotate a vector by an angle θ about axes x, y, z, respectively.

From now on, we will use the corresponding ray $\mathbf{r}$ passing through n and starting at t (subject's 3D head location), as the direction where a subject is looking at, $\mathbf{r} = \{\mathbf{p} = \mathbf{t} + \lambda\mathbf{n} : \lambda \geq 0,\ \lambda \in \mathbb{R}\}$.

Figure 1 shows three subjects looking at the TV with their associated view directions.

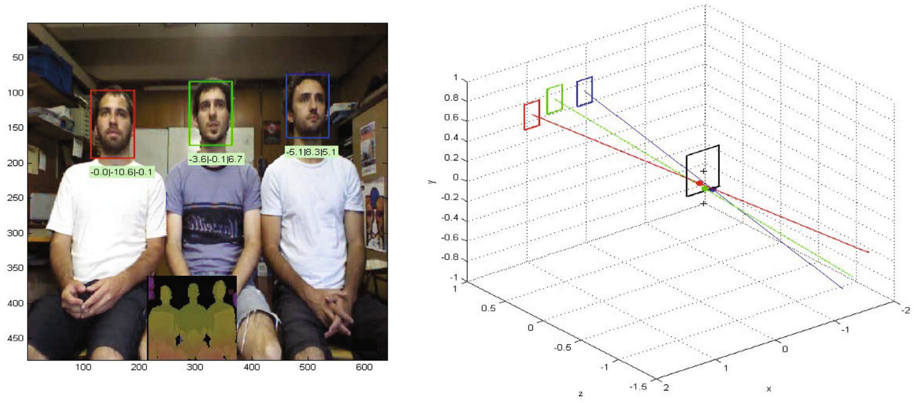

Fig. 1. Left: A color image with its associated depth and people's head pose detected. Right: people's view-direction in a 3D coordinate system centered at the camera. The black square near the origin represents the screen. The units are meters for distances and degrees for rotation angles. Figure best viewed in colors.

3 Camera-Screen Location Estimation

We present an overall algorithm to estimate automatically the camera rotation and screen position with the respect to the camera. We assume that people are looking at the TV and a camera near the TV screen is recording them. The algorithm takes as input an RGB-D video and returns as output the x, y and z coordinates of the screen and also the camera rotation.

The first step of this method is to estimate the TV screen depth, i.e. the z coordinate. Then, once we know the depth distance between the camera and the screen we can further proceed with the estimation of the screen location in x and y coordinates, horizontal and vertical positions, respectively. Finally, an heuristic is employed to infer camera rotation from people's head pose, where we make use of yaw angle to estimate camera rotation. In the next subsections we describe the details of the processes involved in this algorithm.

3.1 Screen Depth Estimation

In order to estimate the distance from the camera to the screen in the z coordinate, we follow the main idea behind multi-view stereo triangulation [7]. Assuming that people look at the same point on the screen when they are watching TV, and are fairly steady and focused, we can potentially estimate the distance from the camera to screen as the intersection between different people's view directions. However, in practice, this requires very good head pose estimation. Thus, we estimate the screen depth as the plane $z = d$ which minimize the pairwise distances between intersection points in d (see Figure 2).

An intersection point p in the plane $z = d$ is obtained as the intersection between the subject's view-direction (see section 2) and the plane. The intersection point is defined as:

$$\mathbf{p} = \mathbf{t} + \lambda_d \quad \text{where} \quad \lambda_d = \frac{d - t_3}{n_3} \tag{1}$$

where λ_d is obtained by derivation of equations for the ray and the plane [7], and $\mathbf{t} = [t_1, t_2, t_3]^T$ and $\mathbf{n} = [n_1, n_2, n_3]^T$, are location and normal vectors of the subject's head, respectively.

For each frame of the RGB-D video, we proceed in the following way:

If there are two or more people, we will have two or more rays for this frame. Since they lie in a 3D space, the rays may not intersect at all. Now let's consider the plane $z = d$ (perpendicular to z axis). Each ray has an intersection point with this plane (see Figure 2). Let $\mathbf{p1}, \mathbf{p2}, \ldots, \mathbf{pm}$ denote m intersection points. Note that when d changes, $\mathbf{p1}, \mathbf{p2}, \ldots, \mathbf{pm}$ will change. We define the average pairwise distance (APD) as:

$$APD(d) = \sum_{i \neq j} ||\mathbf{pi} - \mathbf{pj}||^2 \qquad i, j = 1 \ldots m \tag{2}$$

Because the APD will change as d changes, for each frame f we want to find d_f such that $APD(d_f)$ is the smallest. That is, $d_f = \arg\min_d APD(d)$.

Replacing each point $\mathbf{p1} \ldots \mathbf{pm}$ with its defintion on Equation 1 and solving Equation 2 for d in a least square sense, we can obtain d_f so that $APD(d_f)$ is minimum.

Finally, for the entire video, we compute the screen depth as the average between all d_f estimated in each frame f.

Figure 3 shows the value of $APD(d_f)$ for each frame of an example video where screen is at 0.72 meters from the camera. Note when the APD curve is flat means that people are very steady.

3.2 Screen Location Estimation

To estimate the screen location in x and y coordinates, we assume that screen depth d is already computed. Our main goal is to find where in the plane $z = d$ the screen is actually located.

For this task, we make use of the intersection points in the plane $z = d$ from people's view-directions, in each frame. In this way, we know where in the plane corresponding to the screen, the people are looking at. This will give us, for the entire video, a 2D point cloud in $z = d$. Assuming that when people are watching

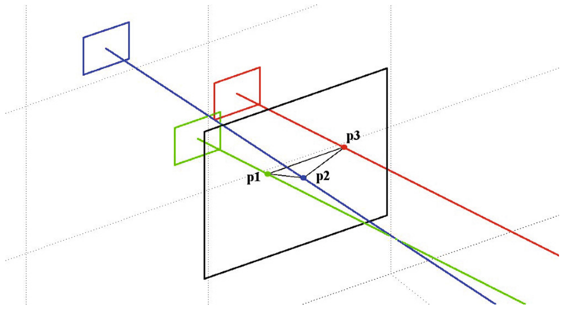

Fig. 2. Example of three rays and its intersections points with a plane, **p1**,**p2** and **p3**. The perimeter of the triangle $\triangle\mathbf{p1p2p3}$ correspond to the *APD* between these points.

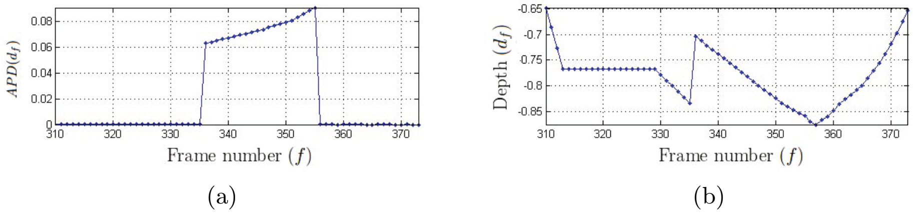

Fig. 3. (a) The minimum of the average pairwise distance between intersection points for each frame. (b) Obtained screen depth for the minimum pairwise distance by frame.

TV are mostly focused on the screen, in the long term, these points will be more concentrated around the center of the screen.

In practice we deal with scattered data and therefore is very hard to determine exactly the center of the screen. So, we simplify the problem in another of finding the vertical (above or below) and horizontal (left, right, center) positions of the screen with respect to the camera. By simply counting how many points fall in each respective quadrant, for a Cartesian coordinate system centered at the camera origin, we will choose the quadrant with the most intersection points, as the screen location in x and y.

Pattern recognition approaches and statistical analyses of the data, such as k-means, also can be employed to infer screen location [3].

3.3 Sensor Rotation Estimation

Following same assumptions as sections 3.1 and 3.2, we estimate the sensor rotation as the average yaw angle of the subject placed closest the camera's center (see Figure 4). As before, this estimation improves as we include more frames and the averages converges to the real sensor rotation.

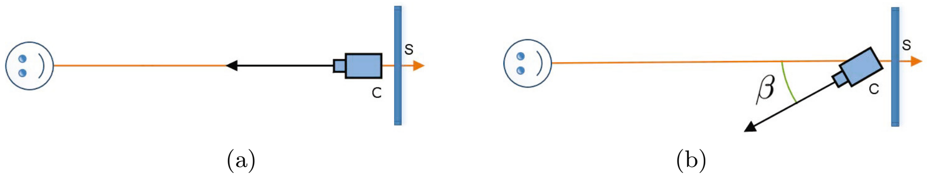

Fig. 4. Subject's yaw angle β coincide with camera C rotation when he or she is looking at the center of the screen S. (a) Yaw angle is zero (b) Yaw angle is β.

4 Screen Localization Validation

In this section we present the behavior of our algorithm for camera-screen localization showing the results of estimating the position of the screen and the camera orientation over a dataset with screen location groundtruth.

People's head poses were obtained with a 3D Face Tracker [2]. Because of camera's RGB and depth poor quality and low resolution head poses tend to be not good enough for our experimental validation. Thus, in order to have good head pose estimates to work with, we checked these sequences in a frame by frame basis, to see if pitch, yaw and roll angles were correct. Then, we manually annotated in each frame if there is a correct value of pitch, yaw or roll. So, in the following subsections we used these verified angles in order to estimate the screen location correctly.

4.1 Dataset Description

In order to verify the process of estimating the position of the screen, several sequences were recorded changing the location of the camera.

The dataset employed consists of 20 sequences in which we simulate a scenario where people are looking at the TV, and there is a camera near the TV recording the people (see Figure 1 for an example). For all the sequences, we placed the camera ahead the screen (varying depth distance), horizontally centered, and with 0 degrees rotation with respect to the plane of the screen. We set the camera in two different heights: 0.78 meters (screen above the camera) and 1.71 meters (screen below the camera).

The length of the sequences varies between 37 and 387 frames. They were acquired at 30 fps and 640x480 resolution for RGB and depth using a Kinect camera. First columns of Table 1 show a detailed summary of each sequence in the dataset: screen vertical position, screen depth, number of frames and number of people.

4.2 Results

Screen location estimation in x and y coordinates give us an overall accuracy of 90 %. Especially, our algorithm in sequences 15 and 17 fail for horizontal estimation and in sequences 8 and 12 fail for vertical estimation. That is to say,

Table 1. Description of each sequence in the dataset and its associated absolute errors in estimation of screen depth and camera rotation

# Seq	Screen Vert Pos	Screen Depth	Frames	# People	Depth Error	Rot Error
1	above	-1.05	387	2	0.08	6.52
2	above	-0.22	239	2	0.10	**17.6**
3	above	-0.22	116	2	0.05	7.99
4	above	-0.55	189	3	0.01	3.19
5	above	-0.55	258	2	0.01	5.63
6	above	-1.05	179	3	0.06	1.88
7	above	-0.29	111	2	**0.12**	2.08
8	above	-0.29	371	2	0.06	2.65
9	above	-0.72	64	3	0.05	1.88
10	above	-0.92	199	2	0.05	0.15
11	above	-1.12	201	3	0.03	1.31
12	above	-1.12	81	3	0.03	4.62
13	above	-0.12	133	2	0.02	2.85
14	above	-0.12	66	3	0.07	0.20
15	below	-0.12	41	2	0.06	3.06
16	below	-0.52	163	2	0.04	**22.16**
17	below	-0.52	37	2	**0.12**	**11.27**
18	above	-0.12	323	3	0.03	5.31
19	above	-0.12	196	3	0.04	1.74
20	above	-0.40	126	3	0.10	0.97

our algorithm returned for 2 of the 20 sequence a wrong value with respect to ground truth, for both the screen vertical and horizontal position, respectively.

Table 1 shows absolute errors, with respect to the dataset groundtruth, when estimating screen depth (i.e. the distance between camera and screen) and camera rotation.

We can see that, screen depth estimation errors in all sequences of the dataset are below 12 centimeters and for the 55 % of the dataset, this error falls below 5 centimeters. These are very promising results for screen depth estimation.

At first sight, camera rotation estimation performance in Table 1 seems not so promising. This is due to, our assumptions for camera rotation estimation (long term data and a subject near the camera's center) are partially met in this dataset. Thus, if we consider sequences 2, 16 and 17 as outliers (error bigger than 10 degrees), the mean error results in 3.06 ± 2.23 degrees, which is a more encouraging outcome.

5 Conclusions and Future Work

We presented a novel method to estimate the TV screen location using a RGB-D camera in a traditional setting where are people watching TV and the camera is looking at them.

For this end, we made several assumptions that can be considered true in the most of the cases found in practice.

The experimental validation gave us promising outcomes that validate our approach in facing this problem and enable us to explore several further applications and improvements.

A possible improvement of this work could be to incorporate the estimation of the TV screen size. Also, if there is only one person in the scene, and he or she changes positions over time, we can potentially estimate the screen depth.

As a general conclusion, we note that with more temporal information, i.e. longer sequences, we can give better estimations.

And as sequences get longer, to incorporate other features like engaging information or focus on attention [1] will be very helpful to filter non useful data and produce accurately results.

We need to remark that our method works as long as the head pose tracking is reasonable and produce good head poses. In the future, while the Moore's law is met, RGB-D sensors will continue improving and our algorithm will be a valuable tool.

Acknowledgments. We would like to thank to Michael Conrad, Geoff Hulten, Kyle Krum, Xu Miao, Yaron Eshet and Beth LaFountaine from the Monroe Team at Microsoft Research for their invaluable help during this project.

References

1. Asteriadis, S., Karpouzis, K., Kollias, S.: Visual focus of attention in non-calibrated environments using gaze estimation. International Journal of Computer Vision, 1–24 (2013)
2. Cai, Q., Gallup, D., Zhang, C., Zhang, Z.: 3D deformable face tracking with a commodity depth camera. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 229–242. Springer, Heidelberg (2010)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. John Wiley & Sons (2012)
4. Fanelli, G., Gall, J., Van Gool, L.: Real time head pose estimation with random regression forests. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 617–624. IEEE (2011)
5. Fanelli, G., Weise, T., Gall, J., Van Gool, L.: Real time head pose estimation from consumer depth cameras. In: Mester, R., Felsberg, M. (eds.) DAGM 2011. LNCS, vol. 6835, pp. 101–110. Springer, Heidelberg (2011)
6. Funes Mora, K., Odobez, J.M.: Gaze estimation from multimodal kinect data. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 25–30. IEEE (2012)
7. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2003)
8. Kondori, F.A., Yousefi, S., Li, H., Sonning, S.: 3d head pose estimation using the kinect. In: 2011 International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–4. IEEE (2011)
9. Murphy-Chutorian, E., Trivedi, M.M.: Head pose estimation in computer vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(4), 607–626 (2009)