Grant agreement no. 297713
Project REWIRE Rehabilitative Wayout In Responsive home Environments
Collaborative project (Small or medium-scale focused research project)
Information and Communication Technologies
ICT-2011.5.1
Small or medium scale focused research project (STREP)

<table>
<thead>
<tr>
<th>Deliverable reference number and title:</th>
<th>D4.3 Final Report and Hand Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date of deliverable:</td>
<td>1st October 2014 - Month 36</td>
</tr>
<tr>
<td>Actual submission date:</td>
<td>1st November 2014</td>
</tr>
<tr>
<td>Start date of project:</td>
<td>1st October 2011</td>
</tr>
<tr>
<td>End date of the project:</td>
<td>31st December 2014</td>
</tr>
<tr>
<td>Organisation name of lead contractor for this deliverable</td>
<td>UOXF (University of Oxford)</td>
</tr>
<tr>
<td>Other organizations involved</td>
<td>UMIL (University of Milan)</td>
</tr>
</tbody>
</table>

Version 1.0

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

<table>
<thead>
<tr>
<th>Dissemination Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU Public</td>
</tr>
<tr>
<td>PP Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE Restricted to a group specified by the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>ISSUE</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Version 0.8</td>
</tr>
<tr>
<td>Version 0.9</td>
</tr>
<tr>
<td>Version 1.0</td>
</tr>
</tbody>
</table>

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies. All rights reserved. The document is proprietary of the REWIRE consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights. This project is supported by funding from the Information Society Technologies Programme under the 7th Research Framework Programme of the European Union.
Contents

1 Introduction ......................................................... 1
  1.1 Some background ........................................... 2

2 Tracking and Calibration using Depth alone .................. 5
  2.1 Image and scene geometry .................................. 5

3 Reconstruction while Tracking a Single Object ............... 11
  3.1 Introduction .................................................. 11
  3.2 Implementation and Evaluation ............................. 15

4 Tracking Multiple Objects ........................................ 19
  4.1 Introduction .................................................. 19
  4.2 Graphical Model .............................................. 19
  4.3 Implementation and Evaluation ............................. 23

5 Conclusions from WP4 ............................................. 29

6 List of publications produced from WP4 ....................... 31

7 Acknowledgments .................................................. 33
Executive Summary

The objective of Work Package 4 in REWIRE has been the development of algorithms for real-time localization and tracking of a patient in rehabilitation, using visual and direct depth sensing. Three guiding principals inform our work. First, the limbs to be tracked, and the contexts and ranges of motion are determined by clinical need, defined in WP3. Second, that the sensing itself and the setting up required are to be applicable in the home environment, rather than in controlled settings of hospital or laboratory. Lastly, that the output be returned at video rate and with small delays, commensurate with driving feedback to the patient-user via the game-playing environment and architecture proposed and developed in WP6.

Deliverable D4.1 (Month 12) was concerned first with understanding the constraints and conditions placed by clinical requirements upon depth and vision sensing; and second with how the chosen sensor, the MS Kinect, could fulfil these conditions both using its standard commercial features and through custom coding. Two particular shortcomings were noted. 1) The standard calibration methods were unable to provide the accuracy required. The deliverable reported methods of calibration of the intrinsics properties of the visual and depth cameras and of calibration of the inter-camera extrinsics. Both methods involved — indeed were highly integrated with — tracking in 3D. 2) Although much of the skeleton was well-recovered by its standard SDK, the Kinect was not able to track and recover the pose of the the lower limbs and feet. The conclusions of D4.1 hinted at the approach to be taken to track these areas of the patient as they performed the game-driven rehabilitation exercise.

Deliverable D4.2 (Month 24), described that response in detail. Two areas were covered: first was that of tracking a known shape – particular here that of the foot; and second is that of recovering that known shape in the first place. For REWIRE it is assumed that the patient is wearing a shoe. Both shape and colour are included in the model. The overall method is a voxel model-based approach with an optimization driven by computation on the level set. We also discuss the issue of pose initialization, and the extension of the tracking method to both feet.

This deliverable, D4.3 (Month 36) describes work intended to allow tracking of multiple objects of highly similar appearance which is applicable both to our earlier work on feet tracking and to hand tracking.

As it is the final report we also take the opportunity in Chapters 2 and 3 to present work from D4.1 and D4.2 within a single framework. The current deliverable extends the framework, and is described in Chapter 4. Conclusions are drawn in Chapter 5 and Chapter 6 gives a list of publications from REWIRE’s Workpackage 4.
Keyword List

Clinician, Rehabilitation, Patient tracking, Visual sensing, Depth sensing, Sensor calibration, Localization, Tracking, Pose Recovery, 3D Object Reconstruction, Multiple objects.

List of Acronyms

2D Two dimensional
3D Three dimensional
CPU Central Processing Unit
CUDA NVidea’s parallel computing platform and programming model
D4.1 Deliverable: Calibration
D4.2 Deliverable: Tracking
D4.3 This deliverable: Hand tracking & Final Reprot
fps Frames per second
GPU Graphics Processing Unit
ICP Iterative Closest Point
IMU Inertial Measurement Unit
RGB-D Red, Green, Blue (ie colour) and Depth
SDK Software Development Kit
WP2 Work Package 2 - User requirements
WP4 Work Package 4 - Patient tracking
WP6 Work Package 6 - Patient Station
1. Introduction

Workpackage 4 in REWIRE has been concerned with the development of real-time localization and tracking of a patient while they rehabilitate after stroke by making movements within a game-playing environment. The sensing modalities used in WP4 are (i) vision and (ii) direct-depth sensing.

Fig. 1.1 illustrates that clinical protocol determines which game a particular patient should play. The game interpreter generates a display involving goals for the patient who is represented on-screen as an avatar. The patient views the screen, and via his or her visual and motor control systems activates limb motion. WP4 closes a feedback loop on the process. RGB-D (Colour-Depth) imagery of the patient is captured, and the poses of the patient and related objects are recovered in real-time, which then drives the avatar’s motion on the screen. Examples of this link are shown in Fig. 1.2.

Deliverable D4.1 was concerned principally with properties of the Kinect RGB-D sensor chosen for the project, and the calibration of its intrinsic and extrinsic parameters to a level sufficient for REWIRE. The method of calibration introduced a method for tracking known objects, but using depth information alone.

Deliverable D4.2 made several contributions, including code packages, and an improved method of silhouette presentation. However the principal achievements were to deliver a method of simultaneously tracking and recovering the structure of unknown objects — specifically hands, shoes and objects with which they are in contact — allowing the building of libraries of known objects. The method used both depth and colour information from the kinect, combining outputs from its two cameras.

In Sections 2 and 3 of this final deliverable D4.3 we will take the opportunity to reprise that work. One of our aims in this final report is to establish an entirely consistent notation throughout. But the main new work in D4.3 is reported in Section 4. It is a

Figure 1.1: Clinical considerations determine the game and level of difficulty in UMIL’s IGER game interpreter. The patient forms the system plant, and is driven by a demand on the display. WP4 supplies feedback to the game engine to drive the display.
method of tracking *multiple* objects of similar appearance, allowing hands and feet to be tracked while imposing the physical constraint that two or more objects should not occupy the same space.

Section 5 draws overall conclusions from WP4, and Section 6 lists all the publications produced. This section continues with a brief review of related work which has informed the approach taken in REWIRE. It is convenient to group our review first around the general theme of model-based 3D tracking and dense reconstruction, before considering the more specialized areas of distance transforms and the tracking of multiple objects with physical constraints.

### 1.1 Some background

#### 1.1.1 Model based 3D tracking

Most existing research work for 3D tracking with or without depth data uses a model-based approach, optimizing the state hypothesis by minimizing some objective function which measures the discrepancy between the expected image cues, cues in conventional and/or depth imagery. While early work of necessity exploited highly sparse features such as points and edges, then different solving tools, such as Bundle Adjustment (e.g. [1]) Kalman Filter(e.g. [2, 3]) or Particle Filter (e.g. [4]) are used for the estimation of object pose. A common algorithm deployed on denser data is Iterative Closest Point (ICP) [5]. In [6], the authors input RGB-D imagery from Kinect and use ICP to track hand-held 3D rigid puppets. The system yields robust and real-time performance, but occlusion introduced by the hand has to be carefully managed through a colour-based pre-segmentation phase. Awkwardly, a different appearance model is required to achieve pre-segmentation when tracking multiple objects. A more general work is KinectFusion [7], where the entire scene structure along with camera pose are estimated simultaneously. Ray-casting is used to establish point correspondence, after which estimation of alignment or pose is achieved with ICP. However, a requirement when tracking with KinectFusion is a static scene, or in our context a single object, a condition which is obviously violated when tracking multiple independently moving objects. Another cluster of RGB-D based dense tracking works use sampling, relying on evaluations of the objective function at many positions in the state space. For exam-
ple, in [8] the author uses Particle Swarm Optimization to track an articulated hand, whereas [9] and [10] use instead a particle filter to estimate pose. Relying on a large number of objective evaluations is computational taxing, even given parallel GPU implementations.

1.1.2 Dense 3D Reconstruction

Most traditional 3D reconstruction methods require a calibrated multiple camera setup and are based on space carving. The introduction of customizable, frame-rate RGB-D cameras has made 3D reconstruction using a single depth camera possible. KinectFusion [7] is among the most successful systems for real-time 3D reconstruction. With a single hand-held Kinect device, KinectFusion can incrementally reconstruct the surface of the physical world that the camera sees, in real-time. However, as discussed in previous subsection, KinectFusion relies heavily on the static world structure to track the camera, thus it can not reconstruct small moving objects. Another related recent work is [11] where the authors use a single, fixed, un-calibrated Kinect to scan human body in a home environment. Accurate 3D human shapes are obtained by combining multiple monocular views of a person moving in front of the sensor. The SCAPE model [12] is used to constrain the alignment of the multiple depth maps from the various views. In [13], the authors use monocular 2D image cues to reconstruct 3D shapes. The reconstruction is constrained by a learnt low-dimensional 3D shape space. Both [11] and [13] are capable of reconstructing 3D objects with a single camera. However, they both rely heavily on a learnt shape space to constrain the reconstruction. This forces the reconstructed objects to be within a fixed, prelearned, category.

1.1.3 Signed distance function for tracking

An alternative to ICP is the use of the signed distance function (SDF). It was first shown in [14] that distance transforms could be used efficiently to register 2D/3D point sets. The SDF was used in [15] to formulate different embedding functions for robust real-time 3D tracking of rigid objects using only depth data, an approach extended in [16] to leverage RGB data in addition. A similar idea is described in [17], where the authors directly use the gradient of the SDF for tracking camera pose. Using only image data, the authors in [18] project a 3D model into the image domain to generate a SDF-like embedding function, and the 3D pose of a rigid object is recovered by evolving this embedding function. KinectFusion [7] also uses a truncated SDF for shape representation, but, as noted earlier, it uses ICP for camera tracking, instead of directly exploiting the SDF. As shown in [17], ICP is less effective for this task.

1.1.4 Tracking multiple objects with physical collision constraints

Physical constraints in 3D object tracking are usually enforced by reducing the number of degrees of freedom (dof) in the state. An elegant example of tracking of always-connected objects (or sub-parts) in this way is given in [19]. However, when tracking multiple independently-moving objects, physical constraints are introduced suddenly and intermittently by the collision of objects: they cannot be conveniently enforced by
dof reduction. Indeed, rather few works explicitly model the physical collision between objects. In [20], the authors track two interacting hands with Kinect input. A penalty term measuring the inter-penetration of fingers is introduced to invalidate impossible articulated poses. In [21, 22], a hand and a moving object are simultaneously tracked, and invalid configurations similarly penalized. In both cases, the measure used is the minimum magnitude of 3D translation required to eliminate intersection of the two objects, a measure computed using the Open Dynamic Engine library [23]. In contrast, in our proposed method, the collision constraint is more naturally enforced though a probabilistic generative model, without the need of an additional constraint engine.
2. Tracking and Calibration using Depth alone

This section reviews the key outcomes of D4.1.

Video material to illustrate the work is available at www.robots.ox.ac.uk/ActiveVision/Research/Programmes/EU_Rewire

2.1 Image and scene geometry

The geometry underpinning work in WP4 is established by describing the calibration of the Kinect sensor using 3D pose tracking, using level sets based on depth input alone. Calibration uses objects of known shape, and does not involve reconstruction.

2.1.1 Image and scene

Each pixel \([x, y]\) in the image \(\Omega_d\) of a depth camera delivers the depth \(Z\) of the associated 3D scene point \(X^c = [X, Y, Z]^\top\) referred to Cartesian coordinates pinned to the depth camera’s optical centre. The projection is perspective, giving the image position \(x = [x, y, 1]^\top\) as

\[
x = Z^{-1}K[I|0]\begin{bmatrix} X^c \\ 1 \end{bmatrix},
\]

and allowing the scene to be recovered from

\[
X^c = K^{-1}Zx
\]

once the intrinsic calibration matrix \(K_{3\times3}\) is known.

Knowledge of the extrinsic calibration between the depth and colour cameras allows the corresponding image point \(x' = [x', y', 1]^\top\) to be found in the colour image \(\Omega_c\) as

\[
\lambda x' = PZx + q,
\]

where \(\lambda\) is a per-point scale, and \(P\) and \(q\) are defined later. The colour 3-vector can then be copied to the depth image as \(c(x, y) \leftarrow c(x', y')\). The pixel at \(x\) in the combined depth-colour image \(\Omega\) then provides \([X^c, c]\).

A pixel is the projection of a scene point that lies either on the background or on the surface of a specific object. An object’s pose relative to the depth camera is represented by a 6-vector \(p\) of modified Rodrigues parameters, from which is easily derived the Euclidean transformation \(T^o_{4\times4}\) relating points in the object and camera frames

\[
\begin{bmatrix} X^c \\ 1 \end{bmatrix} = T^o(p) \begin{bmatrix} X^o \\ 1 \end{bmatrix}.
\]
2.1.2 Tracking of known objects using depth alone

Once calibrated (and the method is reviewed later), the camera delivers a dense set of 3D points, a subset of which should lie on or near to the surface of the object to be tracked.

A typical model-based approach to tracking the object described by a set of points \( \{ X^o_i \} \) on its surface would be, per time step, first to hypothesise the object pose \( p \); then to use \( T^{co}(p) \) to compute expected positions \( \{ \hat{X}^c_i \} \) in the camera frame; and last to minimize some norm of the overall deviation \( \sum_i ||\hat{X}^c_i - X^c_i|| \) from the measured positions. However, this assumes both a neat segmentation and knowledge of the object to scene point correspondences. Neither is available here.

Instead, the object’s surface is defined *implicitly* as the zero-level set \( \Phi(X^o) = 0 \) of an signed distance function (SDF), \( \Phi \). Points \( X^o \) inside the object have \( \Phi < 0 \), and those outside \( \Phi > 0 \). In our work \( \Phi \) has been defined using a signed 3D chamfer distance, the magnitude of which is, for any point \( X^o \), the shortest distance from that point to any point on the object’s surface.

One iteration of our tracking algorithm can be stated as, first, hypothesise a pose \( p \); then transform all the measured scene positions into the object frame using \( T^{oc}(p) \); and last, adjust the pose to minimize a cost function, here made robust using a Geman-McClure M-estimator [24]

\[
E = \sum_{x \in \Omega} \frac{\Phi^2(X^o)}{\Phi^2(X^o) + \omega^2}.
\]

(2.5)

Here, \( \omega \) determines the width of the basin of attraction; and \( x \in \Omega \) and \( X^o \) are related via Eqs. (2.2) and (2.4).

This formulation does not require point to point correspondences, and, further, by placing an upper limit on the per-point chamfer distance (beyond which it is set to zero) the cost is made intrinsically robust without explicit segmentation.

In practice, and as sketched in Fig. 2.1(a), this is achieved by discretizing \( X^o \) in a number of voxels surrounding the object (here, \( 200 \times 200 \times 200 \)). If a transformed point lies

![Figure 2.1](image-url)

Figure 2.1: (a) Objects are defined within a voxelized cube. (b) from right to left: A known object in the depth image, the points in depth camera coordinates and wrapped around the object surface in object coordinates during tracking.
outside all voxels its contribution is ignored. If pixel $x_j$ is back-projected into a voxel, it will be useful to label that voxel as $X^o_j$.

An efficient second order method is used to optimize pose. Voxelization brings further benefit to this process — not only are the per-point chamfer distances pre-computable, but so also are part of their gradients. The cost, Eq. (2.5), is parameterized in terms of a pose change $p^*$ from the current pose, and its gradient found as

$$\frac{\partial E}{\partial p^*} = \sum_{x \in \Omega} \left[ \frac{2\omega^2\Phi}{(\Phi^2 + \omega^2)^2} \frac{\partial \Phi}{\partial X^o} \right] \frac{\partial X^o}{\partial p^*}. \quad (2.6)$$

where $\frac{\partial E}{\partial X^o}$, the term in brackets in Eq. (2.6), is pre-computable.

Levenberg-Marquardt iterations are used to optimize the pose change $p^*$

$$p^* = - \left[ J^T J + \lambda \text{diag} \left[ J^T J \right] \right]^{-1} \frac{\partial E}{\partial p^*} \quad (2.7)$$

where $J$ is the Jacobian matrix of the energy function, and $\lambda$ is the non-negative LM damping factor adjusted at each iteration. The new pose $p_t \leftarrow p^* \otimes p_{t-1}$ is composed with care to preserve the integrity of the rotation.

Fig. 2.2 illustrates output from the tracking process. Ren et al.[15] give results for the performance of this depth-only pose tracker, showing it to run at video rate using just the CPU, and to being robust to significant occlusion.

Figure 2.2: The upper row shows the tracking of a known well-carpentered object using depth data alone, and the bottom row shows a graphics object being added to the video output in the correct pose.
2.1.3 Calibration

So far, we have assumed both intrinsic and extrinsic calibrations are available. The tracking methodology is re-used for calibration.

The intrinsic matrix of the depth-camera has the usual form

\[
K = \begin{bmatrix}
    f_x & s & x_o \\
    0 & f_y & y_o \\
    0 & 0 & 1
\end{bmatrix}.
\]  

(2.8)

To recover the focal lengths, principal point and skew, the optimization’s parameter vector is lengthened to \( \pi = [p^\top k^\top]^\top \) with \( k = [f_x, f_y, x_o, y_o, s]^\top \), and tracking and calibration carried out simultaneously. The cost and gradient have the same form as Eqs. (2.5) and (2.6). Again Levenberg-Marquardt is deployed as the optimizer. Because each video frame \( m \) provides an independent estimate given the current pose of the calibration object, the estimates are combined as

\[
\hat{k} = \sum_{m=1}^{M} [\bar{k} - k_m]^\top \Sigma_m^{-1} [\bar{k} - k_m]
\]  

(2.9)

where the covariance \( \Sigma_m \) at frame \( m \) is approximated by the inverse Hessian. Fig. 2.3(a) shows the calibration object and its being tracked during calibration, and Fig. 2.3(b) shows the convergence of the overall estimate \( \hat{k} \) of the intrinsic parameters as more frames are added, using synthetic data with known ground truth. In low noise, the method recovers the intrinsic parameters accurately from around 200 frames (some 7s of data), while in higher noise the principal point and aspect ratio \( (f_x/f_y) \) are still accurate, but the focal length is biased. In practice, Kinect operates in the low noise regime.

![Intrinsic Calibration](image)

Figure 2.3: Intrinsic calibration. (a) The object. (b) The calibration parameters converging during tracking of the calibration object.
2.1.4 The mapping between cameras

The mapping between colour and depth images (as $P_{3 \times 3} \cdot q_{3 \times 1}$) is found by matching the tracks from point object between cameras. Assuming perspective projection in the RGB camera a scene point $X^c$ in the depth camera coordinates is imaged at

$$\lambda x' = K' [R X^c + t]$$

(2.10)

where $\lambda$ is a per-point scale factor. Using Eq. (2.2) we have

$$\lambda x' = K' R K^{-1} \lambda x + K't = P \lambda x + q.$$  

(2.11)

As illustrated in Fig. 2.4, a wand carrying a characteristically coloured ball is tracked in both cameras to give a set of $x \leftrightarrow x'$ correspondences. For tracking in the colour camera, the ball is detected using a learnt colour model, and a Hough transform applied to the resulting (ball/not-ball) binary mask to determine its centre. To track the ball in the depth camera, the “pose-only” 3D level-set method already described is used, and the ball’s 3D centre recovered. With $M$ pairs of corresponding points, $P$ and $q$ are found by minimizing the 2-norm of the reprojection error

$$[\hat{P}, \hat{q}] = \arg \min_{P,q} \sum_{m=1}^{M} \| x'_m - \hat{x}'_m (P, q, x_m, Z_m) \|^2,$$

(2.12)

where the $x'$ and $\hat{x}'$ here are just 2-vectors. (If needed explicitly, the colour camera’s intrinsic matrix and the extrinsics $R, t$ are found by applying QR decomposition to $[PK]^{-1}$, and so on.)

Figure 2.4: Extrinsic calibration: (a,b,c) the colour image and spherical wand detected and fitted in it; (d,e,f) the depth image and with object detected and fitted.
3. Reconstruction while Tracking a Single Object

This section reviews the key outcome of D4.2, the development of a method of reconstructing the shape of a single object during tracking.

Videos illustrating the work are available at www.robots.ox.ac.uk/ActiveVision/Research/Programmes/EU_Rewire

3.1 Introduction

The ideas underpinning the method introduced of reconstructing while tracking in D4.2 were encapsulated in probabilistic model describing the formation of depth and colour imagery from the scene. An overview of the elements involved is given in Fig. 3.1(a).

It proved convenient to introduce a co-representation of the shape as a set of labelled voxels \( \{X^o, V\} \), where the label takes values \( V = \text{in}, \text{on} \) or \( \text{out} \) depending on the voxel’s being inside, on the surface of, or outside the object. With \( \epsilon > 0 \), \( V \) provides a shorthand for the tests \( \Phi < -\epsilon, |\Phi| < \epsilon, \) and \( \Phi > \epsilon \), where \( \pm \epsilon \) places bounds on being close enough to the surface to be regarded as on the surface.

Fig. 3.1(b) shows the proposed graphical model. The current depth-colour image \( \Omega_t \) depends on the current pose \( p_t \), the voxels and their labels, and the voxels and labels depend on the shape \( \Phi \). The joint distribution can therefore be expressed as

\[
P(\Omega_0 \ldots \Omega_T, p_0 \ldots p_T, \Phi, \{X^o, V\}) = P(\Phi) \prod_i P(X^o_i, V_i|\Phi) \prod_t P(\Omega_t|\{X^o, V\}, p_t) P(p_t),
\]

a form which permits a recursive update of \( \Phi \) and \( \{X^o, V\} \). Two simplifications can be offered immediately. First, observations in the image are taken to be pixel-wise independent, so that

\[
P(\Omega_t|\{X^o, V\}, p_t) = \prod_{x_{jt} \in \Omega_t} P(x_{jt}, e_{jt}|X^o_j, V_j, p_t),
\]

Figure 3.1: (a) Overview of the tracking and reconstruction processes. (b) The graphical model for tracking and reconstruction of a single object.
where $x_{jt}$ back-projects to $X_{jt}^o$. Second, the term $P(X_{jt}^o, V_t|\Phi)$ should properly allow positions $X_{jt}^o$ to be drawn at random from the shape process $\Phi$. In practice, each voxel is addressed once and the probability determined solely from $P(V_t|\Phi)$.

Notwithstanding these simplifications, full inference leading to a MAP estimate of the shape and poses given the imagery appears intractable. Instead, we propose approximate inference at each new frame by alternating phases where (i) the ML estimate of $\Phi$ is found while freezing the camera poses, and (ii) $\mu_t$ is optimized while freezing the shape.

### 3.1.1 Tracking while the shape is frozen

During this phase, the shape $\Phi$ is held constant and we seek a ML estimate of the current pose

$$\max_{\mu_t} P(\Omega_t|\Phi, \mu_t) = \max_{\mu_t} \prod_{x_{jt} \in \Omega_t} P(x_{jt}, c_{jt}|\Phi, \mu_t).$$  \hspace{1cm} (3.3)

Each term in the product can be expanded as

$$P(x_{jt}, c_{jt}|\Phi, \mu_t) = \sum_{v=\text{on, out}} P(x_{jt}|\Phi, \mu_t, V_{jt}=v)P(c_{jt}|V_{jt}=v).$$  \hspace{1cm} (3.4)

Indicator $V=\text{in}$ does not appear here because voxels inside the object are always invisible. The first terms in the sum of products in Eq. (3.4) depend on geometry. Given $\mu_t$, each pixel back-projects to a unique voxel, so that

$$P(x_{jt}|\Phi, \mu_t, V_{jt}=\text{on}) \equiv P(X_{jt}^o|\Phi, V_{jt}=\text{on}),$$  \hspace{1cm} (3.5)

and similarly for $V_{jt}=\text{out}$. But the RHS of Eq. (3.5) is readily expressed qualitatively in terms of the embedding function. $P(X_{jt}^o|\Phi, V_{jt}=\text{on})$ should be sizeable only when $|\Phi(X_{jt}^o)|$ is small — that is, only when $|\Phi(X_{jt}^o)| < \epsilon$ — and vice versa for $P(X_{jt}^o|\Phi, V_{jt}=\text{out})$. These are expressed numerically using a smoothed delta function $\delta^\text{on}$ and a shifted and smoothed Heaviside step $H^\text{out}$:

$$P(X_{jt}^o|\Phi, V=\text{on}) = \delta^\text{on}(\Phi(X_{jt}^o))$$  \hspace{1cm} (3.6)

$$P(X_{jt}^o|\Phi, V=\text{out}) = H^\text{out}(\Phi(X_{jt}^o)).$$  \hspace{1cm} (3.7)

In our implementation, the latter is a logistic function

$$H^\text{out}(\Phi) = \frac{1}{1 + \exp\{- (\Phi - \epsilon)/\sigma_H\}},$$  \hspace{1cm} (3.8)

and probability summing gives

$$\delta^\text{on}(\Phi) = 1 - H^\text{out}(\Phi) - H^\text{in}(\Phi),$$  \hspace{1cm} (3.9)

where symmetry requires $H^\text{in}(\Phi) = H^\text{out}(-\Phi)$. These functions are are plotted in Fig. 3.2(a).
The second terms in Eq. (3.4) come directly from the colour statistics of the object and background. The likelihoods $P(c|V=\text{on})$ and $P(c|V=\text{out})$ are accumulated in 3-channel $\times$ 32-bin histograms, initialized either from an object detector or from a user-selected bounding box on the depth-colour image, in which the foreground model is built from the interior of the bounding box and the background from the immediate region outside the bounding box (as in Fig. 3.1(a)). Once initialized, the histograms are updated continuously while tracking.

Combining the two terms, we obtain a likelihood of the image as a function of the pose

$$L(p_t) \sim \prod_{j:x_j \in \Omega_t} \left\{ P_{\text{on}}(c_j|\Phi(X_{j'}^o)) + P_{\text{out}}(c_j|\Phi(X_{j'}^o)) \right\}$$

(3.10)

where $P_{\text{on}}(c_j|\Phi(X_{j'}^o))$ is shorthand for $P(c_j|V_j'=\text{on})$, and so on. This is written as a summed cost $\mathcal{E}$ by taking logs, and optimized using Levenberg-Marquardt. The derivatives are

$$\frac{d\mathcal{E}}{dp^*} = \sum_j \left\{ \left( \frac{P_{\text{on}}(\Phi(X_{j'}^o)) + P_{\text{out}}(\Phi(X_{j'}^o))}{P(x_j, c_j|\Phi, p)} \right) \frac{\partial \Phi}{\partial X_{j'}^o} \frac{\partial p^*}{\partial X_{j'}^o} \right\}.$$  

(3.11)

### 3.1.2 Reconstruction while the poses are frozen

For the reconstruction phase a foreground depth-colour image $\hat{\Omega}_t$ is created from $\Omega_t$ by including only those pixels for which $P_{\text{on}}(c_j|V_j'=\text{on}) > P_{\text{out}}(c_j|V_j'=\text{out})$ currently. The objective then is to maximize over $\Phi$

$$P(\Phi|\hat{\Omega}_{1...t}, p_{1...t}) \propto P(\hat{\Omega}_{1...t}|\Phi, p_{1...t}) P(\Phi) =$$

$$\sum_{v=\text{in, out}} P(\hat{\Omega}_{1...t}|\{X_i^o, V=v\}, p_{1...t}) P(\Phi|V=v) P(V=v).$$

(3.12)

Note that we allow the shape to be determined by voxels with $v=\text{in}$ and $v=\text{out}$ alone: the number of the voxels on the object surface is negligible by comparison.

The first term on the RHS of Eq. (3.12) gives, per voxel, the likelihood of generating the measured pixel’s depth value. Given a particular voxel $X_i^o$ and the current pose $p_t$,
a pixel position is determined from \( \hat{x} = K^T \mathbf{p}_t | X^o_i | \). The closest measured pixel \( x \) is determined and back-projected into the scene, allowing a fractional signed range discrepancy \( d \) to be determined along the direction of the projected ray as

\[
d_i = \frac{[R(p_t)X^o_i + t(p_t)]^T[K^{-1}Zx]}{|K^{-1}Zx|^2} - 1 ,
\]

which is positive if the voxel \( X^o_i \) lies behind the measured surface, and vice versa. The resulting likelihoods of lying in or out of the object are approximated using

\[
L_{\text{in}}(i) = 0.5(\text{sgn}(d_i)e^{-|d_i|/\sigma_d} + 1) ,
\]

\[
L_{\text{out}}(i) = 1 - L_{\text{in}}(i) ,
\]

forms that are equivocal \( (L_{\text{in}}, L_{\text{out}} \approx 0.5) \) about voxels that are either far from the surface or exactly on the surface, but give a low probability of being inside if the voxel is immediately in front of the surface, and vice versa. \( L_{\text{in}} \) is plotted in Fig. 3.2(b). (This form of likelihood does not increase \( L_{\text{out}} \) if the surface is visible behind the voxel. Neglecting that extra information has not been detrimental in our experiments.) The time-cumulative likelihood \( L_{1:t}(i) \) for a particular voxel \( i \) is found as

\[
L_{1:t}(i) = L_{t}(i)L_{1:t-1}(i) .
\]

Returning to Eq. (3.12), the second term is expressed per voxel by the shifted and smoothed Heaviside steps given earlier

\[
P(\Phi(X^o_i)|V=\text{out}) = H_{\text{out}}(\Phi(X^o_i)) ,
\]

\[
P(\Phi(X^o_i)|V=\text{in}) = H_{\text{in}}(\Phi(X^o_i)) .
\]

The third term, the prior \( P(V) \), plays an important role in stabilising the estimation because any given depth-colour image only yields information about voxels between the camera and the object, and nothing about those behind the surface of the object. The prior is defined with respect to a chosen shape (ie. SDF) \( \Phi_o \):

\[
P(V_i=\text{in}) \equiv P(X^o_i, V_i=\text{in}) = G(\Phi_o(X^o_i)) ,
\]

\[
P(V_i=\text{out}) = 1 - G(\Phi_o(X^o_i)) .
\]

In all our work we use as prior

\[
G(\Phi) = a(1+e^{\Phi/\sigma_G})^{-1} + (1 - a)(0.5) , \quad a \in [0, 1]
\]

with influence \( a=0.5 \) and smoothness \( \sigma_G=4 \), as shown in Fig. 3.2(c). An uninformative prior has \( a=0.0 \).

Combining the three terms, we have per voxel

\[
P(\Phi(X^o_i)|\Omega_{0...t}, P_{0...t}) \sim
\]

\[
L_{1:t}(i) [1-H_{\text{in}}(\Phi(X^o_i))] G(\Phi_o(X^o_i)) + L_{1:t}(i) H_{\text{in}}(\Phi(X^o_i)) [1-G(\Phi_o(X^o_i))] ,
\]
from which a total probability is found as a product, assuming per voxel independence. The cost to optimize is found by summing over logs, and the shape $\Phi$ found using gradient flow methods, as is standard for level sets. In the implementation we also add a regulariser that encourages the gradient of the level set to have unit magnitude, preserving the SDF nature of $\Phi$ [25].

### 3.2 Implementation and Evaluation

#### 3.2.1 Implementation

The tracking module has been implemented on an Intel Core i7 3.4GHz CPU, where with known or already-reconstructed objects it executes in around $10\text{ ms}$ per frame. The more intensive reconstruction module is implemented on an Nvidia GTX 680 GPU, where each iteration takes around $1\text{ ms}$. Starting from an uninformative prior we find that the first frame requires 100 to 200 iterations to obtain the shape of the observed object with sufficient fidelity to permit tracking into the next frame. After that, the reconstruction refinement can proceed with significantly fewer iterations per frame (30 is typical), resulting in an overall processing time per frame of $35\text{ ms}$.

#### 3.2.2 Evaluation

We begin with several real-world tracking and reconstruction sequences, which show that our method is robust to initialization error and outliers and can work in unconstrained environments. Figs. 3.3(a, b) show reconstruction while tracking over a few hundred frames of a rigid hand and a shoe. Both were initialized as cubes. The last column shows views around the 3D reconstruction itself. The experiments were performed in an uncontrolled environment, and the objects were small and moving — they could not be reconstructed by KinectFusion for example.

Fig. 3.4 shows an evaluation of tracking performance using ground truth data obtained from synthesized depth-colour imagery. A virtual RGB-D camera was moved around a known object model, covering all viewpoints, and reconstruction initialized as a sphere. After each frame the recovered pose was compared with ground truth. The translational accuracy is simply the Euclidean distance between the estimated and ground truth poses. The rotational accuracy is represented by the average included angle $r_{\text{err}} = \frac{1}{3} \sum_{i \in \{x,y,z\}} \cos^{-1} (\hat{\mathbf{e}}_i^T \hat{\mathbf{e}}_i^\text{g})$ between three fixed axis directions $[\hat{\mathbf{e}}_x \hat{\mathbf{e}}_y \hat{\mathbf{e}}_z]$ after rotation by the ground truth matrix $R_\text{g}$ and by its estimate $R_\text{e}$. The plots show results from two tests on the generated RGB-D sequences. The first assumes the 3D shape of the object is known and tests only 3D tracking, while for the second the object shape is both reconstructed and tracked. The tracking error with a known object is less than $2\text{ deg}$ in rotation and 1 pixel unit in translation, values which rise to $8^\circ$ and 4 pixel units when also reconstructing.
Figure 3.3: Stills cut from sequences showing stages during reconstruction while tracking a hand (a), and a shoe (b). The top image of each pair shows the colour image and that below shows the reconstruction result overlayed. Missing depth data is shown in green.

Figure 3.4: Quantitative evaluation of the precision our method for tracking 3D rigid object on synthetic data. The error in translation is measured in pixel unit while rotation is measured in degree.
The next experiment compares our tracking and reconstruction with that of KinectFusion [7]. Since KinectFusion requires a static scene, we mount the object (an L-shaped piece of sponge) above a desktop and move the Kinect around it, recording the sequence using the Kinect’s SDK. Sample frames are shown in Fig. 3.5(a). The first row shows the colour frames, the second row visualizes our reconstruction result projected onto the images (aligned with depth frame), and the third row shows the KinectFusion reconstruction result up to current frame — the “fusion frame” from the SDK. Fig. 3.5(b) compares the final reconstruction results of our method and KinectFusion. Both methods produce a visually correct result. The 3D model produced by KinectFusion has an inaccurate lip on the top surface, while that part is correctly reconstructed by our algorithm. Our method does however produce a more noisy surface below the reconstructed 3D shape, in the areas that have never been observed by the camera. This is because noisy outlier depth pixels can propagate incorrect membership probabilities to areas in the 3D volume where the related views of object has never be observed. The similarity in quality is also evident in Fig. 3.5(c) where we compare the camera poses recovered by the two approaches. Our pose tracking result is very close to the output of KinectFusion, despite relying only on the local geometry of the reconstructed object rather than on all of the wider scene. (The KinectFusion camera pose was obtained directly from the SDK using 384³ voxels to fill 1 m³. The inevitable fixed Euclidean transform between the two sets of poses has been removed in the figure by aligning the trajectories of two camera centres.)

Lastly we explore the sensitivity to the tuning parameter $\sigma_d$ in Eq. (3.15), which controls the thickness of the reconstructed model. Fig. 3.6 shows the impact on reconstruction accuracy as $\sigma_d$ is changed over an order of magnitude from 4 to 40. Again, synthetic imagery was generated to provide ground truth. The object had a notional volume of 100 mm³ and reconstruction was initialized from a sphere of radius 50 mm. Fig. 3.6 shows that the initial average alignment error was 6 mm. When $\sigma_d < 20$, the error decreases as more frames are observed and converges at around frame 150, i.e. after some 5 seconds. When $\sigma_d = 20$, even though the average alignment error does not converge to the 2 mm value achieved earlier, the reconstructed shape is still visually correct, though larger than the real object. When $\sigma_d = 40$, the reconstructed shape is poorly reconstructed after the first few frames — it is much too thick — resulting in tracking failure in all following frames. The “thickness” that $\sigma_d$ controls is relative to the size of the volume that contains the 3D level-set embedding function. As noted earlier, we use 200³ voxels to capture $\Phi$ in our implementation. Large objects are scaled and reconstructed in the same fixed volume. In experiments with different objects in various of environments we find that $\sigma_d = 8$ provides good reconstruction and tracking performance. This value was used in all tests reported earlier.
Figure 3.5: A comparison between our method and KinectFusion for 3D reconstruction. (a) Four sample frames. The first row shows the colour frame (aligned into the depth camera frame), the second projects our reconstruction onto the image, and the third shows the KinectFusion fusion frame. (b) Our final reconstruction compared with KinectFusion. (c) Quantitative comparison between camera pose output when using KinectFusion [7] and our method. Translation is measured in mm while rotation is measured in degrees.

Figure 3.6: Quantitative evaluation of the accuracy of our method for 3D reconstruction with different $\sigma_d$ in Eq. (3.15).
4. Tracking Multiple Objects

This section describes novel work which forms the principal contribution to deliverable D4.3.

Video material to illustrate the work is available at www.robots.ox.ac.uk/ActiveVision/Research/Programmes/EU_Rewire

4.1 Introduction

To pursue the problem of hand tracking in REWIRE, and to improve the quality of foot tracking, in D4.3 we have considered the problem of tracking multiple instances of the models already reconstructed using the methods of Section 3.1.

Multi-object 3D tracking involves the recovery of $M$ sets of sequences of multiple poses $\hat{p}_t = \{p_1 \ldots p_M\}_t$ given the $M$ object models $\{\Phi_1 \ldots \Phi_M\}$ and the depth-colour imagery $\{\Omega_1 \ldots \Omega_t\}$.

In particular we consider the challenging problem of tracking objects of similar appearance. Although more difficult in absolute terms, one point of mathematical simplification is that the objects share the same colour appearance model. As in Section 3.1, only two such models are needed to describe the colour statistics of the scene, $P(c|V=\text{on})$ for the objects, and $P(c|V=\text{out})$ for the background. As before, these are initialized either from an object detector or from a user-selected bounding box, and updated continuously while tracking.

4.2 Graphical Model

The graphical model underpinning our multi-object tracking algorithm is shown for a particular timestep in Fig. 4.1(a). Because there are multiple object frames, it proves convenient to consider voxel locations in the camera frame, making their values dependent on the pose. To make the generative process more intuitive, we introduce an intermediate variable $\Phi^c$, which is the union all 3D object shapes in the camera frame. In the model, the locations of voxels $X^c$ are treated as generated randomly from the shape $\Phi^c$, though all voxel locations in camera coordinates have the same probability of being generated. Later, at Eq. (4.11), we will emphasize that in practice the voxels remain in the object frame and are not actually duplicated in the camera frame. Indeed, when $M=1$ there is no difference between the operation of this model and that of Section 3.1.

The objective is to find the optimal temporal set $\{\hat{p}_0 \ldots \hat{p}_t\}$ of the poses of $M$ objects given the set of object shapes $\Phi_1 \ldots \Phi_M$ and observed depth-colour imagery $\{\Omega_1 \ldots \Omega_t\}$,

$$\max_{\hat{p}_0 \ldots \hat{p}_t} P(\hat{p}_0 \ldots \hat{p}_t | \Phi_1 \ldots \Phi_M, \Omega_0 \ldots \Omega_t) \quad (4.1)$$
Figure 4.1: (a) Graphical model for our multi-object tracker. (b) Illustration of the fusion of multiple object SDFs and projection process. SDFs are first transformed into camera coordinates then fused together by a minimum function. The observed RGB-D image domain is generated by the fused SDF.

The set of \( M \) poses at time \( t \) is optimized by maximizing the posterior of pose

\[
P(\mathbf{\hat{p}}_t|\Phi_1 \ldots \Phi_M, \Omega_t) \sim P(\Omega_t|\Phi_1 \ldots \Phi_M, \mathbf{\hat{p}}_t)P(\mathbf{\hat{p}}_t|\Phi_1 \ldots \Phi_M) \tag{4.2}
\]

as a function of \( \mathbf{\hat{p}}_t \). In the following subsections, we will refer to the image likelihood \( P(\Omega_t|\Phi_1 \ldots \Phi_M, \mathbf{\hat{p}}_t) \) as the data term and the pose prior term \( P(\mathbf{\hat{p}}_t|\Phi_1 \ldots \Phi_M) \) as the physical constraint term.

### 4.2.1 Data fitting term

Omitting the subscript \( t \) and introducing the intermediate variable \( \Phi^c \), the first or data term in Eq. (4.2) is

\[
P(\Omega|\Phi_1 \ldots \Phi_M, \mathbf{\hat{p}}) = P(\Omega|\Phi^c)P(\Phi^c|\Phi_1 \ldots \Phi_M, \mathbf{\hat{p}}) \tag{4.3}
\]

Assuming that the observations are pixel-wise independent, then \( P(\Omega|\Phi^c) \) can be decomposed into a product of per-pixel likelihoods

\[
P(\Omega|\Phi^c) = \prod_j P(\mathbf{x}_j, c_j|\Phi^c) \tag{4.4}
\]

Given the “shape union” \( \Phi^c \), and marginalizing \( V \), the per-pixel likelihood becomes

\[
P(\mathbf{x}_j, c_j|\Phi^c) = \sum_{v=\text{on}, \text{out}} \{ P(\mathbf{x}_j|\Phi^c, V_{j'}=v)P(V_{j'}=v|c_j) \} \tag{4.5}
\]

Using the reasoning developed earlier, the pixel location likelihoods for the foreground and background are distributed as

\[
P(\mathbf{x}_j|\Phi^c, V_{j'}=\text{on}) = \delta^\text{on}(\Phi^c(\mathbf{X}_j^c)) \tag{4.6}
\]

\[
P(\mathbf{x}_j|\Phi^c, V_{j'}=\text{out}) = H^\text{out}(\Phi^c(\mathbf{X}_j^c)) \tag{4.7}
\]

Thus the likelihood of the image given the shape union varies as

\[
P(\Omega|\Phi^c) \sim \prod_{j: \mathbf{x}_j \in \Omega} \left\{ \delta^\text{on}(\Phi^c(\mathbf{X}_j^c))P^\text{on}_{c_j} + H^\text{out}(\Phi^c(\mathbf{X}_j^c))P^\text{out}_{c_j} \right\} \tag{4.8}
\]
where, as before, $P_{c}^{v_{j}}$ is shorthand for $P(c_{j}|V_{j}=v)$, all at timestep $t$. The difference between the image likelihood here and in Eq. (3.10) is that the likelihood is no longer a direct function of the $M$ object poses $\hat{p}$. Instead, $\Phi$ is interjected as a function of $M$ object poses and object shapes $\{\Phi_{1} \ldots \Phi_{M}\}$.

To proceed we require an expression for $\Phi_{c}(X^{c})$ for any $X^{c}$. Given a set of object shapes in their respective object frames $\{\Phi_{1} \ldots \Phi_{M}\}$ and their corresponding set of poses $\{p_{1} \ldots p_{M}\}$, each can be transformed (notionally) into the camera frame as $\{\Phi_{c1} \ldots \Phi_{cM}\}$.

By definition, the value at a point of the combined SDF $\Phi_{c}$ is the minimum value of the individual SDFs at that point

$$\Phi_{c}(X^{c}) = \min \left(\Phi_{c_{1}}(X_{c}^{c}), \Phi_{c_{2}}(X^{c}), \ldots, \Phi_{c_{M}}(X^{c})\right).$$

(4.9)

In practice we use a analytical relaxation

$$\Phi_{c}(X^{c}) = -\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp \{-\alpha \Phi_{c_{m}}(X^{c})\}$$

(4.10)

where $\alpha$ controls the smoothness of the approximation. Although large $\alpha$ gives better approximation of the minimum function, in experiment we find that using a smaller $\alpha$ gives a wider basin of convergence for our tracker.

We note that there is no need copy the voxels in object frame into the camera frame to evaluate Eq. (4.10). For a particular $X^{c}$ and object $m$, $X^{o}$ is found from $[X^{o}]^{T} = T_{oc}(p_{m})[X^{c}]^{T}$ and the SDF is

$$\Phi_{m}(X^{c}) \equiv \Phi_{m}(X^{o}).$$

(4.11)

Thus the image likelihood in Eqs. (4.3,4.8) becomes:

$$P(\Omega|\Phi_{1} \ldots \Phi_{M}, \hat{p}) \sim$$

(4.12)

$$\prod_{j:x_{j} \in \Omega} \left\{ \delta_{on} \left( -\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp \{-\alpha \Phi_{m}(X^{c}_{j})\} \right) \right. \left. P_{c}^{on} \right.$$  

$$+ H_{out} \left( -\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp \{-\alpha \Phi_{m}(X^{o}_{j})\} \right) P_{c}^{out} \right\}.$$  

The negative log of Eq. (4.12) provides the data-fitting cost $E_{data}$ of the overall cost $E$. We differentiate this energy term w.r.t. the set of pose parameters $\hat{p} = \{p_{1}, \ldots, p_{M}\}$ in which each $p_{m}$, recall, is a 6-vector. Then

$$\frac{\partial E_{data}}{\partial \hat{p}} = \sum_{j:x_{j} \in \Omega} L_{j} \frac{\partial \Phi^{c}(X^{c}_{j})}{\partial \hat{p}},$$

(4.13)

where in detail

$$L_{j} = \frac{\delta^{on}'(\Phi^{c})P_{c}^{on} + H_{out}'(\Phi^{c})P_{c}^{out}}{\delta_{on}(\Phi^{c})P_{c}^{on} + H_{out}(\Phi^{c})P_{c}^{out}},$$

(4.14)

$$\frac{\partial \Phi^{c}(X^{c}_{j})}{\partial \hat{p}} = -\frac{1}{\alpha} \sum_{m=1}^{M} w_{m} \nabla \Phi_{m} \frac{\partial X^{o}_{j,m}}{\partial \hat{p}},$$

(4.15)
and
\[ w_m = \frac{\exp\{-\alpha \Phi_m(X^o_{j',m})\}}{\sum_{k=1}^M \exp\{-\alpha \Phi_k(X^o_{j',k})\}}. \tag{4.16} \]

Here \( \delta^{on'} \) and \( H^{out'} \) are derivatives of \( \delta^{on} \) and \( H^{out} \) respectively. Pixel \( x_j \) has a back-projection \( X^c_{j'} \) in the camera frame which in turn is transformed to \( X^o_{j,m} \) in the \( m \)-th object’s coordinate frame. The row vector of spatial gradients of the SDFs are computed using central differences in each object’s frame
\[ \nabla \Phi_m = \left[ \frac{\partial \Phi_m}{\partial X}, \frac{\partial \Phi_m}{\partial Y}, \frac{\partial \Phi_m}{\partial Z} \right] \tag{4.17} \]

Since the shape of the object is known from the outset, these gradients can be found in advance.

The derivative of \( E_{data} \) has a very clear meaning. Given a pixel \( x_j \) in the depth-colour image domain, instead of assigning this pixel deterministically to a certain object, we back-project it into camera coordinates and thence into each objects’ coordinates with the current set of object poses \( \hat{p} \). Then, a membership weight \( w_m \) is computed as in Eq. (4.16). If the back-projection \( X^o_{j,m} \) is close to the \( m \)-th object’s surface \( (\Phi(X^o_{j,m}) \to 0) \) but far from other objects, then \( w_m \to 1 \) and the other membership weights tend to zero, preserving \( \sum w_m = 1 \). Thus \( w_m \) can be interpreted as the probability that a pixel \( x_j \) originated from the \( m \)-th object.

### 4.2.2 Collision constraint term

The second likelihood term in Eq. (4.2) is decomposed into a product of per-pose probabilities
\[ P(\hat{p}|\Phi_1 \ldots \Phi_M) = P(p_1|\Phi_1 \ldots \Phi_M) \times \prod_{m=2}^M P(p_m|\{p_1 \ldots p_{m-1}\}, \Phi_1 \ldots \Phi_M), \tag{4.18} \]

but as we do not impose pose priors on objects we can ignore \( P(p_1|\Phi_1 \ldots \Phi_M) \).

The remaining products can be used to enforce pose-related constraints. Here we use them to avoid collisions in the tracker by defining them so that a surface point on one object is discouraged from moving inside any other object.

For each object \( \Phi_m \) we uniformly and sparsely sample a set of \( K \) collision points \( C_m = \{C^o_{m1} \ldots C^o_{mK}\} \) from its surface in object coordinates. At each timestep these are transformed into the camera frame as \( \{C^c_{m1} \ldots C^c_{mK}\} \) using the current pose \( p_m \). Let us denote the partial union of SDFs \( \{\Phi^c_{1} \ldots \Phi^c_{m-1}\} \) by \( \Phi^c_{-m} \). Our proposition now is that we can write
\[ P(p_m|\{p_1 \ldots p_{m-1}\}, \Phi_1 \ldots \Phi_M) \sim \frac{1}{K} \sum_{k=1}^K H^{out} \left( \Phi^c_{-m}(C^c_{m,k}) + \xi \right) \tag{4.19} \]
where $H^\text{out}$ is the smoothed Heaviside function. Ignoring $\xi$, the rationale is that if all the collision points on object $m$ lie \textit{outside} the shape union of objects 1 to $m-1$, this quantity will tend to unity, whereas as more and more of the collision points lie inside the quantity tends to zero. The offset $\xi$ is a small positive value that prevents probability of objects being very close becoming too small.

The negative log-likelihood of Eq. (4.19) gives us the second part of the overall cost

$$\mathcal{E}_{\text{coll}} = -\sum_{m=1}^{M} \log \left( \frac{1}{K} \sum_{k=1}^{K} H^\text{out}\left( \Phi^c_{m-}(C^c_{m,k}) + \xi \right) \right).$$

(4.20)

4.2.3 Optimization

The overall cost is the sum of the data term and the collision constraint term $\mathcal{E} = \mathcal{E}_\text{data} + \mathcal{E}_\text{coll}$. We use the local frame to evaluate the derivative at identity at each iteration, and use Levenberg-Marquardt (LM) to compute the incremental change in all poses together (6m parameters). The “multi-pose” increments are then as in Eq. (2.7)

$$\ddot{p}^* = - \left[ J^T J + \lambda \text{diag} [J^T J] \right]^{-1} \frac{\partial \mathcal{E}}{\partial \dot{p}}.$$  

(4.21)

4.2.4 Online learning of appearance model

The object/background appearance model $P(c|V)$ is again important for the robustness of the tracking, and we again adapt the appearance model online after the tracking is completed on each frame. However the process is changed slightly. As before we use pixels in the immediate surrounding region of the objects to compute the background model, but we use pixels that have $\Phi^c(X^c) \leq 3$ to compute the surface appearance model. This involves points that best fit the surfaces of \textit{multiple} objects. The online update of appearance model is achieved by using a linear opinion pool with learning rates $\{\rho_{\text{on,out}}\}$:

$$P_t(c|V_i = v) = (1 - \rho_v^v)P_{t-1}(c|V_i) + \rho_v^v P_t(c|V_i).$$

(4.22)

In all our experiments we set $\rho_{\text{on}} = 0.05$ and $\rho_{\text{out}} = 0.3$.

4.3 Implementation and Evaluation

4.3.1 Implementation

The multi-object tracker has been implemented on an Intel Core i7 3.4GHz CPU, where a 30Hz frame rate is maintained with up to 5 objects. A version specialized to two objects has been coded on an Nvidia GTX 680 GPU for application to tracking the feet and hands of patients in rehabilitation. However, for just two objects that fill a small fraction of the depth-colour image it turns out that the time per frame of around 15ms is very similar to that of the CPU. Only when there are more and larger objects is the computational power of the GPU fully exploited.

Fig. 4.2 shows the processing time for the CPU implementation as the number of objects, each of which is of similar complexity, is increased. As expected, the method is $O(M)$. 
4.3.2 Evaluation

Quantitative

We illustrate the multi-object tracker’s performance with two sets of quantitative experiments. In the first we use synthetic data to compare the tracking accuracies of the multi-object tracker with two instances of the single object tracker of Section 3.1. Two objects of known shape are moved periodically towards and away from each other. Gaussian noise is added to both the rendered colour and the depth images. Four sample frames from the test sequence are shown in Fig. 4.3(a). In Fig. 4.3(b) the green line shows the relative distance between the two objects, but the value has been scaled and offset for visualization. The blue and red curves show the error in pose for the multi-object and single object trackers, respectively. The errors in translation and rotation are represented using the method of Section 3.2.2. It can be seen that when the two objects with similar appearance are neither overlapping nor close (eg.frame 94), both tracking methods provide accurate results. However, when the two objects move close together the single object trackers produce a large error by comparison. The single object tracker fails to model correctly the pixel membership, which leads to an incorrect pixel association. The soft pixel membership in the multi-object tracker solves this problem.

The second quantitative experiment makes a similar comparison, but with real imagery. In the absence of ground truth for the absolute pose, we instead measure the consistency of the relative pose between two static objects while moving the camera around them. Example frames are shown in Fig. 4.4(a). When the two recovered poses are accurate one would expect consistent relative translation and rotation through the whole sequence. As shown in Fig. 4.4(b), our multi-object tracker is able to recovered much more consistent relative translation and rotation than two of the independent single-
Figure 4.3: (a) Sample synthetic RGB-D frames used in our experiment, frame number correspond to the marks in (b). (b) A comparison of pose estimation error between our multi-object tracker and two instances of the single-object tracker of Section 3.1.

Figure 4.4: (a) Sample depth-colour frames used in our experiment. (b) Comparison of the variance in relative pose estimation between our multi-object tracker and two instances of the single-object of Section 3.1.

Object trackers.

Qualitative

We use four sequences to illustrate the robust performance of our multi-object tracker. Fig. 4.5(A) shows the tracking of two pieces of sponge with identical shape and appearance. Panel (a) in shows colour and depth images along with the per-pixel foreground probability $P_{c}^{on}$. Panel (b,top) shows above the per-pixel membership weight $w_{i}$ (from Eqn. 4.16) during tracking, where the magenta and cyan colours correspond to the two objects and the blue coloured pixels having ambiguous membership; and panel (b,bottom) shows the final tracking result. The tracker is able to track accurately through heavy occlusion and to handle challenging motions.

In Fig. 4.5(B) we show similar outputs from simultaneously tracking a white cup and a white ball. This demonstrates the effectiveness of the physical collision constraint. The cup provides both occlusion and physical constraint to the ball. Even though there is no depth observation from the ball in this experiment (owing to significant occlusion from the cup), the method can still accurately estimate the location of the ball using the physical constraint alone.

Fig. 4.6 shows a challenging multi-object sequence where five identically yellow coloured, but differently shaped toy bricks are tracked. The top sequence shows a close up of the...
Figure 4.5: (A) Film strip showing the multi-object algorithm tracking two pieces of cut foam. (a) shows typical RGB imagery along with the per-pixel foreground probability \( P_{\text{fg}} \), and (b) shows five stills from the video with (top) snapshots of cut foam. (A) shows typical RGB and Dimagery along with the per-pixel foreground probability \( P_{\text{fg}} \), and (b) shows five stills from the video with (top) snapshots of cut foam. (A) shows typical RGB and Dimagery along with the per-pixel foreground probability \( P_{\text{fg}} \), and (b) shows five stills from the video with (top) snapshots of cut foam. (A) shows typical RGB and Dimagery along with the per-pixel foreground probability \( P_{\text{fg}} \), and (b) shows five stills from the video with (top) snapshots of cut foam. (A) shows typical RGB and D
original colour imagery and the bottom sequence shows the tracked objects reprojected onto the imagery in red. In spite of the heavy self-occlusion and the occlusion included by hands, our tracker can still track robustly and accurately.

When tracking in real-world scenarios, it is often difficult to obtain accurate 3D object models. In Fig. 4.6(B) we show our tracker working with approximate 3D shapes reconstructed using the method of Section 3.1. Fig. 4.6(B) we track two interacting feet with a pair of coarse shoe models. Throughout most of the sequence our tracker successfully recovers the two poses. However, we do also encounter two failure cases here. The first one is visible in column 4 of Fig. 4.6(B), where the shoe is incorrectly rotated. This failure is caused by the 3D model’s partial rotational ambiguity around its major axis. The second failure case can be seen in column 6, where a black shadow causes the ground pixels to have a high foreground probability (most clearly seen on the 3rd row). With the majority of one foot occluded, the tracker incorrectly tries to fit the shoe model to the ground pixels with high foreground probability. As an indication of its robustness, our tracker automatically recovers from both failure cases.
tracking result. The tracker failed on the frames of column 3 and 5 but recovered by column 6.

The film strips show the challenging sequence where 5 pieces of toy bricks of identical yellow colour are tracked. The top row shows the original colour images, and the lower row projects and renders in red the five tracked objects. The top row shows the per-pixel membership weight — magenta and cyan correspond to the two objects, and blue coloured pixels are ambiguous. The bottom row shows the tracking result. The tracker failed on the frames of column 3 and 5 but recovered by column 6.
5. Conclusions from WP4

Within WP4 in REWIRE, a framework has been developed that uses 3D level-set functions to solve 3D tracking and reconstruction problems from combined depth and colour imagery, and which allows accurate tracking of feet and hands in the context of rehabilitation.

The geometry underlying the framework was introduced in D4.1 and has been reprised in Section 2 of this final report in the context of sensor calibration. Tracking from depth alone allowed the time-varying pose of known objects to be recovered, sufficient to map that pose onto objects within REWIRE’s game environment.

With calibrated depth-colour imagery available, two probabilistic models with developed to exploit it.

In D4.2, the first probabilistic model permitted simultaneous tracking and reconstruction of a single 3D object whose shape was previously unknown. The implemented method has been shown able to track moving objects in quite arbitrary surroundings at 100Hz, and has been found robust to occlusion and missing data in the depth-colour frames. The reconstruction module allows 3D voxelized models of previously unseen objects to be built in real-time. Reconstruction evolves a 3D level-set embedding function on a per-voxel inside/outside probability volume, which is learned incrementally. The probabilistic formulation allows the imposition of a 3D shape prior onto the shape evolution, which in turn permits initialization of the tracking-reconstruction pipeline with simple 3D shapes. The shape optimization involves independent per-voxel operations and has been implemented in a massively parallel fashion on a GPU.

In D4.3 we have devised a second probabilistic model for tracking using depth and colour imagery. Although the model assumes already reconstructed 3D objects, it allows the tracking of multiple objects with identical appearance as a collective single entity. There are several advantages over instantiating multiple individual trackers, not least that naturally obtain weights that indicate the probability of individual image observations being generated by each of the tracked objects, thus implicitly solving the data association problem. Furthermore, the formulation naturally leads to a physical constraint term which allows us to specify prior knowledge about the world. Here this term has been used to indicate that it is highly unlikely that more than one object occupies the same location in 3-space.

We note that the approach to tracking and reconstruction from RGB-D developed during REWIRE represents the state of the art in this area. Although less advanced solutions might have sufficed for simple tracking parts of the project, the approach builds in a methodological robustness.

As the dense tracking and reconstruction framework is region-based and uses sim-
ple histograms as appearance models, it is particularly well suited for broadly untex-
tured objects such as hands and feet (and most shoes). It is easy to imagine scenarios
where richer appearance models might be better suited. A direction of research beyond
REWIRE would be to explore texture-based appearance models.
6. List of publications produced from WP4

1. A Unified Energy Minimization Framework for Model Fitting in Depth
   C Y Ren and I D Reid
   Proc 2nd ECCV Workshop on Consumer Depth Cameras for Computer Vision (CDC4CV)
   May 2012, pp 72-82

2. Dense Reconstruction Using 3D Object Shape Priors
   A Dame, V A Prisacariu, C Y Ren and I D Reid
   Proc 17th Int Conf on Computer Vision and Pattern Recognition, San Francisco, CA, June 2013

3. Robust Silhouette Extraction from Kinect Data
   M Pirovano, C Y Ren, I Frosio, P L Lanzi, V Prisacariu, D W Murray and N A Borghese
   Proc 17th Int Conf on Image Analysis and Processing, Sept 11-13, Naples, Italy, 2013

4. Simultaneous 3D Tracking and Reconstruction on a Mobile Phone
   V A Prisacariu, O Kahler, D W Murray and I D Reid
   Proc Int Symp on Mixed and Augmented Reality, Oct 1-4 2013, Adelaide, Australia

5. STAR3D: Simultaneous Tracking And Reconstruction of 3D Objects Using RGB-D Data
   C Y Ren, V A Prisacariu, D W Murray and I D Reid
   Proc Int Conf on Computer Vision Dec 3-6 2013, Sydney, Australia

6. Regressing Local to Global Shape Properties for Online Segmentation and Tracking
   C Y Ren, V Prisacariu and I D Reid

7. 3D tracking of multiple objects with identical appearance using RGB-D input
   C Y Ren, V A Prisacariu, O Kahler, D W Murray and I D Reid
   To be presented, Int Conf on 3D Vision, 3DV 2014, Tokyo, Dec 8th-11th, 2014

8. Real-Time 3D Tracking and Reconstruction on Mobile Phones
   V A Prisacariu, O Kahler, D W Murray and I D Reid
   In press (Oct 2014) IEEE Transactions on Visualization and Computer Graphics

9. 3D reconstruction and tracking of objects from depth-colour data
   C Y Ren, V A Prisacariu, D W Murray and I D Reid

10. A Probabilistic Approach to Real-time reconstruction and tracking from Depth-Colour Imagery
    C Y Ren
7. Acknowledgments

The Oxford team wishes first to acknowledge the support of the EU’s FP7 programme, and second to acknowledge and thank Professor Alberto Borghese at the University of Milano for leading the project with limitless energy, skill and patience.

As current Principal Investigator at Oxford it is also my pleasure to acknowledge the planning and forethought put in to the proposal for REWIRE by Prof Ian Reid, now at the University of Adelaide. Ian has continued to be involved in much of the work.

He and I also record our debt of gratitude to Dr Victor Prisacariu and doctoral student Carl Ren for their tireless work in executing the work in WP4. Carl has recently submitted his DPhil thesis which is based largely on the work reported here and in previous deliverables.

David Murray
Active Vision Lab
Oxford
19th Oct 2014
Bibliography


