Detecting Deficient Coverage in Colonoscopies

Daniel Freedman, *Member, IEEE*, Yochai Blau, Liran Katzir, Amit Aides, Ilan Shimshoni, Danny Veikherman, Tomer Golany, Ariel Gordon, *Member, IEEE*, Greg Corrado, Yossi Matias, *Member, IEEE*, and Ehud Rivlin

***Abstract*—Colonoscopy is tool of choice for preventing Colorectal Cancer, by detecting and removing polyps before they become cancerous. However, colonoscopy is hampered by the fact that endoscopists routinely miss 22-28% of polyps. While some of these missed polyps appear in the endoscopist's field of view, others are missed simply because of substandard coverage of the procedure, i.e. not all of the colon is seen. This paper attempts to rectify the problem of substandard coverage in colonoscopy through the introduction of the C2D2 (Colonoscopy Coverage Deficiency via Depth) algorithm which detects deficient coverage, and can thereby alert the endoscopist to revisit a given area. More specifically, C2D2 consists of two separate algorithms: the first performs depth estimation of the colon given an ordinary RGB video stream; while the second computes coverage given these depth estimates. Rather than compute coverage for the entire colon, our algorithm computes coverage locally, on a segment-by-segment basis; C2D2 can then indicate in real-time whether a particular area of the colon has suffered from deficient coverage, and if so the endoscopist can return to that area. Our coverage algorithm is the first such algorithm to be evaluated in a large-scale way; while our depth estimation technique is the first calibration-free unsupervised method applied to colonoscopies. The C2D2 algorithm achieves state of the art results in the detection of deficient coverage. On synthetic sequences with ground truth, it is 2.4 times more accurate than human experts; while on real sequences, C2D2 achieves a 93.0% agreement with experts.**

***Index Terms*—Colonoscopy, coverage, depth estimation, unsupervised deep learning.**

I. Introduction

Colorectal Cancer (CRC) is a global health problem, resulting in an estimated 900K deaths per year [1]; it is the second deadliest cancer in the United States [2].

Manuscript received March 19, 2020; accepted April 24, 2020. Date of publication May 21, 2020; date of current version October 28, 2020. *(Corresponding author: Daniel Freedman.)*

Daniel Freedman, Yochai Blau, Liran Katzir, Amit Aides, Ilan Shimshoni, Danny Veikherman, Tomer Golany, Yossi Matias, and Ehud Rivlin are with Google Research, Haifa 31905, Israel (e-mail: danielfreedman@google.com; yochaib@google.com; lirank@google.com; amitaides@google.com; ishimshoni@google.com; veikherm@google.com; tomergolany@google.com; yossi@google.com; ehud@google.com).

Ariel Gordon and Greg Corrado are with Google Research, Mountain View, CA 94043 USA (e-mail: gariel@google.com; gcorrado@google.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMI.2020.2994221

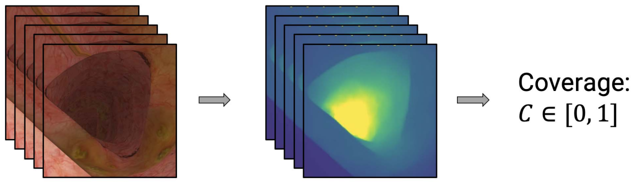

Fig. 1. Our algorithm computes a depth image from a given RGB image. Then, based on the computation of a depth image sequence from a video sequence, the algorithm can compute local coverage, and therefore detect where the coverage has been deficient and a second look is required.

CRC is different from other leading cancers in that it is preventable. Specifically, polyps, which are small precancerous dwellings in the colon, may be detected and removed before they actually become cancerous. Colonoscopy is considered the gold standard procedure for the detection and removal of polyps. Whereas fecal immunochemical and related tests may detect CRC once it has become malignant, colonoscopy is able to detect the polyps in their precancerous stage, thereby preventing cancer from developing. And in contrast to wireless capsule endoscopy, colonoscopy can not only detect, but also remove, polyps.

Unfortunately, the literature indicates that endoscopists miss on average 22-28% of polyps during colonoscopies, which includes 20-24% of adenomas [3]. (An adenoma is a polyp which has the potential to become cancerous; this is in contrast to a hyperplastic polyp, which is benign.) There is therefore room for improvement in polyp detection during colonoscopies. The importance of these missed polyps can be quantified in terms of the rate of interval CRC, defined as a CRC that is diagnosed within 60 months of a negative colonoscopy [4]. In particular, it is estimated that a 1% increase in the Adenoma Detection Rate (ADR, defined as the fraction of procedures in which a physician discovers at least one polyp) can lead to a 6% decrease in the rate of interval CRC [5].

It is therefore imperative to decrease the polyp miss-rate during colonoscopies. There are several factors which lead endoscopists to miss polyps. Some factors, such as bowel preparation, can only be addressed by better patient compliance with the preparatory process. But other factors, such as endoscopist fatigue and endoscopist experience level, can be aided by AI-based real-time decision support systems. In particular, given a well-prepped bowel, there are two principal reasons why an endoscopist might miss a polyp: (1) the polyp appears in the field of view, but the endoscopist misses it, perhaps due to the difficulty of detection, e.g. the polyp may be very small or flat; (2) the polyp does not appear in the field of view, as the endoscopist has not properly covered the relevant area during the procedure. Note that these two reasons are orthogonal, and demand different types of computer vision-based solutions. In terms of the first reason, polyp detection systems such as [6], [7] have been shown to be quite effective. In this paper we choose to focus on the second reason for missing polyps: deficient coverage.

As we have explained, our main motivation for computing coverage is to detect when said coverage is deficient, and thereby decrease the polyp miss-rate. There is a secondary motivation, however, which is that coverage is a performance measure, similar to ADR, by which endoscopists can be graded. The consensus within the field of gastroenterology is that for a procedure to be effective, 90-95% of the colon ought to have been covered [8]. An algorithm for computing coverage could therefore be used both for alerting the endoscopist to missed regions, as well as for measuring the endoscopist's performance.

We refer to our approach to coverage computation as the C2D2 algorithm: Colonoscopy Coverage Deficiency via Depth. C2D2 consists of two separate algorithms. The first performs depth estimation of the colon given an ordinary RGB video stream; while the second computes coverage given these depth estimates. We now outline each of these in turn. The method we use for depth estimation is based on a deep learning approach, in which the network maps RGB images directly to depth images. One advantage presented by any network-based solution is that it allows for the depth estimation algorithm to run in real-time. However, the particular deep learning approach we use offers two further benefits. First, the approach relies only on unsupervised data; thus, one can learn directly from colonoscopy videos without the need for any supervisory signal. Alternative techniques are often based on learning from synthetic data, for which there is depth-supervision, e.g. [9], [10]; however, this entails the need for domain adaptation, which we avoid. Second, our method is calibration-free: it learns the camera intrinsics as part of the algorithm. This is particularly important, as acquiring the intrinsic parameters of a given endoscope is not straightforward; and each colonoscopy will use a different endoscope, entailing different parameters.

Given the depth estimates, C2D2 can then compute coverage, and detect when it is deficient. Coverage is computed on a segment-by-segment basis; we will make the definition of coverage precise in Section IV, but for now, it may be thought of as a scalar in [0, 1] which measures what fraction of the colon has been viewed in any given segment. The coverage algorithm is also based on deep learning, but due to the particular character of the problem – that is, the impossibility of ground truth labelling on real colonoscopy sequences – we must train on synthetic sequences. However, in the final analysis the coverage algorithm must also work on real sequences. The joint requirements of training on synthetic sequences but inference on real sequences leads to a novel two network architecture, with a corresponding two stage training process.

To the best of our knowledge, the C2D2 algorithm is the first to be evaluated on a large scale test set (previous work has tended to perform evaluation on a handful of examples e.g. [11]). We provide quantitative performance results on a collection of 561 synthetic sequences with ground truth. Our results show that on this set, C2D2 outperforms physicians by a factor of 2.4, according to the Mean Average Error (MAE) of coverage. On real sequences, no ground truth is available. Instead, we show that on a set of 301 real sequences, C2D2 achieves a 93.0% agreement with human experts. We also provide qualitative performance results on real sequences, which show that C2D2 outputs highly plausible coverage scores that agree with the eyeball test.

These results demonstrate the value of the C2D2 system: the computation of coverage in general, and detection of deficient coverage in particular, are highly geometric tasks. In such tasks, it is often the case that computers outperform humans, and this is borne out by our results. In many AI tasks, the goal is simply to do as well as human experts; in our case, the system outperforms humans, and this is where its true value lies.

To summarize, our contributions are fourfold:

1) We propose a novel approach to coverage, which is implemented using a two network architecture with a corresponding two stage training process.
2) We present the first calibration-free unsupervised method for depth estimation applied to colonoscopies.
3) The combined C2D2 system is the first coverage system to be evaluated in a large-scale way, and outperforms human experts by a wide margin on coverage tasks.
4) We release a dataset of synthetic colonoscopy videos on which C2D2 was trained and evaluated.

The remainder of the paper is organized as follows. Section II reviews related work, focusing on coverage and depth estimation within endoscopic procedures; as well as more general modern techniques for SLAM and visual odometry. Section III presents our technique for calibration-free, unsupervised learning of depth estimation. Section IV defines precisely the coverage problem we would like to solve, and describes our algorithm for tackling this problem. Section V presents results for both depth estimation as well as coverage, including a detailed description of the new coverage dataset we have collected. Section VI concludes.

II. Related Work

We begin by reviewing the three papers which are, each in a given aspect, most related to the current work. In [12] and its follow-up paper [13], Wu *et al.* propose a blind-spot detector for the EGD (esophagogastroduodenoscopy) procedure, which is an endoscopic procedure focusing on the upper GI tract, including the pharynx, esophagus, stomach, and duodenum. The idea is, in some sense, to provide a notion of coverage of the upper GI tract; the goal is therefore similar to the goal of the current paper. Wu *et al.* divide the upper GI tract into 26 areas, and devise a CNN-based per-frame detector to classify a given image according to which of the 26 regions it belongs to. A technique based on reinforcement learning is built on top of this classifier in order to encourage temporal consistency. Wu *et al.* then verify the usefulness of the real-time system in a randomized controlled trial, and show that endoscopists using the system experience far fewer blind-spots (i.e. regions which are not viewed) than those not using the system, 5.86% vs. 22.46%. The main difference between the approaches of Wu *et al.* and the current work concerns the notion of coverage which is proposed: in Wu *et al.*, coverage is in terms of semantic regions, whereas our approach has a much more *geometric* notion of coverage. We argue that the colon does not have as many varied or differentiated areas as the upper GI tract: the colon, which is quite long, is generally divided into 6 different regions – the cecum, the ascending colon, the transverse colon, the descending colon, the sigmoid colon, and the rectum. Therefore a semantic approach to coverage would not work nearly as well in the case of the colon, as the regions are simply too large. A geometric approach, by contrast, allows for an area of any given size (even a single frame) to be analyzed in terms of coverage, and is therefore a considerably more flexible approach.

A second related work is that of Ma *et al.* [11], which focuses on the problem of depth reconstruction in the colon. The reconstruction pipeline proposed by [11] is complex, but essentially consists of two pieces: a deep network piece, which computes both the depth image for the current frame as well as the camera pose; and a more traditional set of SLAM-based geometric procedures for refining the depth and pose, and for stitching the depth images together to create a 3D point cloud. It is important to note that the network is trained in a supervised fashion; as colonoscopy videos do not come with depth ground truth, a proxy for the ground truth is computed using a separate (non-deep) Structure from Motion algorithm [14]. They then use the 3D reconstruction to provide a measure of coverage. The distinctions between the approach of Ma *et al.* and our approach are twofold. First, the depth pipeline of Ma *et al.* requires supervised data, whereas our technique is purely unsupervised. Supervision based on Structure from Motion is an interesting idea, but it is difficult to know how effective it is, given that the evaluation in the paper also assumes that the Structure from Motion is the ground truth; it may therefore be that the pipeline has simply learned to compute Structure from Motion depth estimates, rather than true depth. Second, and more importantly, the coverage algorithm proposed in the paper is not thoroughly evaluated. Rather, missing region fractions are simply quoted for four colon segments, with the numbers verified by a colonoscopist. In the current work, we provide a full-scale evaluation of the proposed coverage algorithm.

A final piece of closely related work is the paper of Turan *et al.* [15], which takes an unsupervised approach to visual odometry and depth reconstruction in the colon, with the primary application being robotic endoscopic capsules. The approach is based on one of the early unsupervised deep learning techniques for visual odometry and depth estimation for the general computer vision audience [16]. The differences between our work and this work are fourfold. First and most importantly, our work does not require known camera intrinsics to work, that is, it is calibration-free. This is a major difference, as each endoscope has its own intrinsics which are generally not simple to compute. Second, our work is based on a very recent technique for unsupervised depth estimation [17], which has been shown to have superior accuracy to [16] on the KITTI dataset. Third, Turan *et al.* do not provide any evaluation of their depth estimation algorithm, instead simply showing a few images (they focus instead on evaluating the odometry part of the algorithm). Fourth, Turan *et al.* do not discuss issues of coverage.

Other related work has focused on somewhat different versions of the depth reconstruction problem. Turan *et al.* [10] make use of a supervised deep learning pipeline quite similar to that introduced in [9] for performing visual odometry and depth reconstruction. They are most interested in visual odometry, which they evaluate on a dataset they have collected for motion within a porcine stomach. Both Chen *et al.* [18] and Rau *et al.* [19] use supervised approaches to depth estimation, where the supervision comes from a synthetic dataset; both use adversarial techniques to ensure that the predicted depth resembles true depth images. In the case of Chen *et al.* [18], they further perform stitching on the depth images to yield a single unified point cloud, using the ElasticFusion technique [20]. Widya *et al.* [21] perform 3D reconstruction on the whole stomach. Their technique is based on extraction of SIFT features, followed by a classical Structure from Motion approach [14], [22]. SIFT features are generally known to be problematic in medical images, but the technique works here due to the use of chromoendoscopy, in which the stomach itself is dyed using indigo carmine. Nevertheless, the reliance on dyeing severely limits the applicability, as chromoendoscopy is not very common. Slightly older work includes [23], which computes a two-dimensional visibility of the colon; [24] which is an offline (non-real time) technique that uses classical techniques based on Shape from Shading and Shape from Motion to produce a dense 3D reconstruction; and [25], which bases its colon surface reconstruction on the geometry of the Haustral ridges. Finally, we mention a trio of works [26]–[28] whose purpose is to compute pure visual odometry (i.e. pose) without regard to either depth estimation or coverage.

We conclude by briefly surveying the literature related to recent techniques in SLAM, visual odometry, and depth reconstruction intended for the broader computer vision audience. Earlier work, such as [29]–[31], was geometric in character, and did not use any sort of learning pipeline. Initial work which applied deep learning techniques, including [9], [32]–[34], did so using the supervision available in such datasets as KITTI. More recent work [16], [17], [35]–[37] has moved to unsupervised deep learning approaches, and is therefore broadly applicable wherever one has video sequences; no depth images are required. We will go into greater detail regarding one of the most recent (and most successful) of these unsupervised techniques [17] in Section III.

III. Calibration-Free Unsupervised Depth Estimation

We are interested in learning how to estimate a sequence of depth images directly from the corresponding sequence of RGB images of the colonoscopy procedure. We would like to take a deep learning approach to this problem. The standard way of tackling this problem requires supervision, in the form of a depth image for each RGB image; for example, such data exists in the case of the KITTI dataset [38], and is sometimes acquired by equipping the capture device with a depth sensor in addition to a regular camera. In the case of endoscopy, several such datasets exist, including those used in [39] based on both CT and dense stereo; those used in [40] based on CT; and those used in [41], based on stereo, structured illumination, and time of flight.

In this study, instead, we turn to a purely unsupervised approach to depth estimation. Over the last two years, a series of papers on deep learning of unsupervised depth estimation have appeared in the computer vision literature [16], [17], [36], [42]–[44]. All of these papers are based on essentially the same principle, and differ in their details. The principle is the *view synthesis loss*, which we now explain.

A. General Algorithmic Approach

We proceed as follows. Instead of trying to solve the problem of unsupervised depth estimation, which seems to be hard, we try to solve an even harder problem: we try to estimate both the depth image and the pose of the camera (sometimes called the visual odometry) simultaneously. Solving a harder problem seems to be counter-intuitive, but it will afford us an extra benefit in that we can tie the depth and the pose together. Specifically, we define the pose as the rigid transformation (rotation and translation) from the current frame t to the previous frame $t-1$. In particular, we imagine that we have two separate networks that we learn: the depth network takes as input the current RGB frame, and outputs the corresponding depth image; while the pose network takes as input the current and previous RGB frames, and outputs the pose. See Figure 2. Given this setup, we imagine the following series of steps:

- We take the current RGB image I_t, and pass it through the depth network to get the current depth image D_t.
- We take the current and previous RGB frames I_t and I_{t-1}, and pass them through the pose network to get the pose, expressed as a rotation matrix R and translation vector t.
- Considering the depth image D_t as a point cloud, we transform each of the points into the previous frame, according to the standard formula:

$$z'p' = KRK^{-1}zp + Kt, \tag{1}$$

In the above, p and p' are the original and transformed homogeneous coordinates of the pixel, respectively; z and z' are the original and transformed depth of the pixel, respectively; and K is the intrinsic camera matrix:

$$K = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

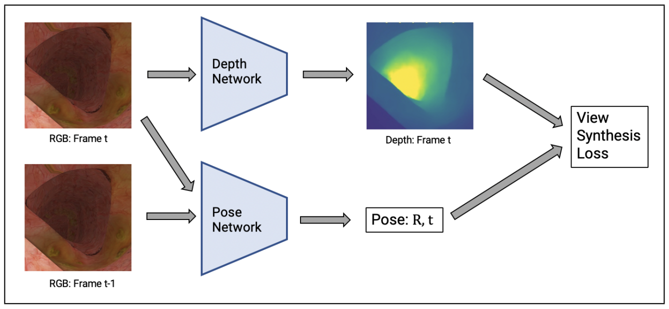

Fig. 2. The view synthesis loss and corresponding network architecture. See accompanying description in the text.

- Given the projected 3D points, one can then re-render the points using the original RGB values at I_t, to get a projected RGB image in the $t-1$ frame, which we label $\hat{I}_{t-1}$.
- If the depth and pose have been computed correctly, the original RGB image at $t-1$ and the new projected image $\hat{I}_{t-1}$ ought to be equal! Thus, our loss is given by $\delta(I_{t-1}, \hat{I}_{t-1})$, where δ is some metric between images, e.g. L_1. This is the view synthesis loss.

B. Elimination of the Need for Calibration

The view synthesis loss and the corresponding network architecture is illustrated in Figure 2. One issue, as can be seen by examining Equations (1) and (2), is that the camera needs to be calibrated prior to using this technique. This poses a problem in our case, as each endoscope model (of which there are many) has its own set of intrinsics, and we cannot rely on the possibility of endoscope calibration. This problem is inherent in most of the work on unsupervised monocular depth estimation, e.g. [16], [35]–[37]. However, following the very recent technique [17], we can predict not only depth and pose but also the camera intrinsics (K) as well. This necessitates relatively minor changes to the network architecture: in addition to the depth and pose subnets, there is also a camera intrinsics subnet. Despite the relative simplicity of implementation, this change is crucial in practice, allowing us to deploy the depth estimation pipeline on any endoscope.

C. Details and Caveats

Regarding the image metric δ between $\hat{I}_{t-1}$ and I_{t-1}, we use two separate metrics: the L_1 difference and the structural similarity (SSIM). In addition to RGB consistency, depth consistency is enforced through an L1 penalty on the difference between the warped depth at the source pixel (z') and the native depth at the target frame. We use the same mechanism as in [17] to avoid enforcing consistency in areas that become occluded or disoccluded upon transitioning between the two frames.

There are two important caveats. First, in this work we assume that motion is caused primarily due to camera motion. Under this assumption, t and R are the same for all the pixels in the entire frame, and Equation (1) maps every pixel in a source frame to a target location on the target frame. In doing so, we are neglecting the non-rigid deformations of the colon. However, if the non-rigid deformations are sufficiently small between any two frames, this is a reasonable approximation. Given that the video is taken at 30 fps, meaning that only 33 ms separates two frames, this approximation may hold in practice. In any case, the results seems to bear out the use of this simplified model.

The second caveat is that the depth image is correct only up to a scale factor, i.e. a single scale factor for the entire image. In principle, the scale factor that is effectively returned by the algorithm should be arbitrary, but in practice, this factor seems to be fairly consistent across long video segments. From the point of view of coverage, the more critical point is that whatever degree of arbitrariness remains in the scale factor, the coverage algorithm learns to deal with effectively.

IV. The C2D2 Algorithm: Computing Coverage

When considering colon coverage, the natural goal is to estimate the fractions of covered and non-covered regions of a complete procedure. Such a formulation of the problem is useful for the physician in terms of a retrospective analysis of a given procedure, as well as general guidance for future procedures. A more interesting goal, however, is the real time estimation of coverage fraction, on a segment by segment basis; that is, while traversing the colon, the goal is to estimate what fraction of the *current* segment has been covered. The implications of such a functionality are clear: during the procedure itself, the physician may be alerted to segments with deficient coverage, and can immediately return to review these areas. This in turn ensures that a higher proportion of polyps will be seen.

We begin this section with a formulation of the coverage problem, including the precise definition of what is meant by segment coverage. We then discuss our overall approach to the problem, which is based on a two-stage training procedure using synthetic data. We then discuss each of the stages in turn. The first stage is a per-frame computation, in which visibility is computed for a given frame. The second stage takes the network learned in the first stage, and uses it in order to learn per-segment coverage, which is our ultimate goal.

A. Formulation of the Problem

We begin by defining the coverage in the colon in a mathematically consistent fashion. A 3D model of a colon consists of the pair $(\mathcal{M}, s)$ where:

- $\mathcal{M}$ is a 3D mesh forming the surface of the colon.
- $s(\cdot)$ is a 3D curve, $s : [0, L] \to \mathbb{R}^3$, traversing the whole colon, and lying in the center of the colon. The curve is parameterized by its distance ℓ along the curve from the beginning of the colon (rectum). This curve is known as the *lumen* of the colon.

We can associate to each point m on the mesh $\mathcal{M}$ the closest point to it on the lumen and its corresponding parameter value:

$$\ell^*(m) = \underset{\ell \in [0,L]}{\arg\min} \|m - s(\ell)\|$$

Similarly, a given camera position p within the colon can also be associated to its nearest point on the lumen, and for ease of notation we also denote the corresponding parameter value as $\ell^*(p)$.

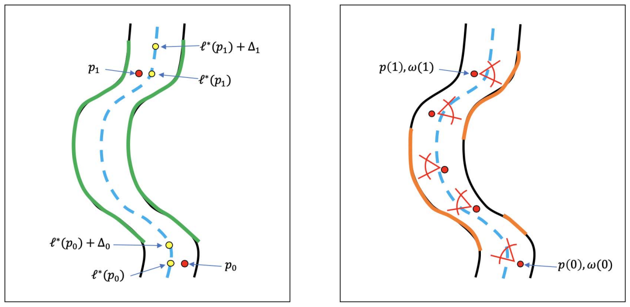

Fig. 3. Illustration of coverage. In both figures, the colon is shown in 2D in black, and the lumen is the dashed blue curve; camera locations are shown as red dots. Several points on the lumen are denoted in yellow, along with their corresponding parameter values $\ell \in [0, L]$. Left: the maximal set of visible points $\mathcal{V}(p_0, p_1)$ is shown as the green curves, which are a subset of the colon surface (black). Right: for a given trajectory, illustrated by red camera locations and viewing angles (at a discrete set of points which subsample of the trajectory), the set of actually visible points $\mathcal{A}(p(\cdot), \omega(\cdot))$ is shown in orange. The coverage is then the ratio of the area of the orange points to the area of the green points.

Now, consider a segment of a colonoscopy video, where the initial and final camera positions are p_0 and p_1. Assuming that the path the camera takes is monotonic – that is, the camera is moving from the end of the colon (the cecum) towards the rectum pointing in the direction of the cecum – then the maximal set of points on the colon that can be visible is given by

$$\mathcal{V}(p_0, p_1) = \{m \in M : \ell^*(p_0) + \Delta_0 \leq \ell^*(m) \leq \ell^*(p_1) + \Delta_1\} \tag{3}$$

In the above, Δ_0 accounts for the viewing angle of the camera: due to the fact that the camera has a field of view that is less than 180°, one cannot see details that are immediately to the side of the initial camera location p_0. Hence $\Delta_0 > 0$; and as the viewing angle becomes smaller, Δ_0 becomes larger. By contrast, Δ_1 accounts for the fact that the image taken from the deepest point on the sequence can view deeper points on the colon. Specifically, the deepest points are ones whose closest point on the lumen is a further distance Δ_1 from the closest point on the lumen to the final camera position p_1. These concepts are illustrated in the left subfigure of Figure 3.

The above computation deals with the maximal set of visible points. In practice, not all points are viewed, and this is what leads to deficient coverage. Specifically, given a particular camera position $p \in \mathbb{R}^3$ and orientation $\omega \in \boldsymbol{\Omega}$, we can define the actual set of points on $\mathcal{M}$ that are visible, which we denote $\mathcal{A}(p, \omega)$. This is computed by rendering the image given the mesh and the camera pose (position and orientation), given the camera's internal calibration parameters (focal length and principal point); one can then verify which points on $\mathcal{M}$ appear in the rendered image, and these are in the points in $\mathcal{A}(p, \omega)$. Given a full camera trajectory, which we denote by $p : [0, 1] \to \mathbb{R}^3$ and $\omega : [0, 1] \to \boldsymbol{\Omega}$, the set of actually visible points for the whole trajectory is simply given by

$$\mathcal{A}(p(\cdot), \omega(\cdot)) = \bigcup_{t \in [0,1]} \mathcal{A}(p(t), \omega(t)) \tag{4}$$

These concepts are illustrated in the right subfigure of Figure 3.

Finally, given a particular camera trajectory $(p(\cdot), \omega(\cdot))$, the coverage is defined as the ratio of actually visible points to maximally visible points. That is, combining Equations (3) and (4), we define the coverage as

$$\mathcal{C}(p(\cdot), \omega(\cdot)) = \frac{\mu[\mathcal{A}(p(\cdot), \omega(\cdot))]}{\mu[\mathcal{V}(p(0), p(1))]} \tag{5}$$

where μ is the standard measure. It is important to note that using the standard measure implies that coverage is based on the fraction of surface area, rather than the fraction of pixels. Note also that in practice, if the vertices on the mesh are sufficiently dense, then one can simply count the vertices in both $\mathcal{A}(p(\cdot), \omega(\cdot))$ and $\mathcal{V}(p(0), p(1))$.

We end this section by noting that it is common practice in colonoscopy screening for the endoscopist to retroflex the endoscope during withdrawal to examine proximal sides of folds and closely examine the mucosa. Effectively, this means that the endoscopist examines one side of the colon wall, immediately followed by an examination of the other side of the wall. A natural question might be: how does this affect the definition of coverage in Equation (5)? The answer is that the definition of coverage can accommodate this situation without difficulty. The video segment in question contains both sides of the wall, which implies that the set of actually visible points $\mathcal{A}(p(\cdot), \omega(\cdot))$ contains both sides of the wall, and is therefore equal to (or nearly equal to) the set of maximally visible points $\mathcal{V}(p_0, p_1)$. Therefore, the coverage $\mathcal{C}(p(\cdot), \omega(\cdot))$ will be equal to (or nearly equal to) 1, as desired.

B. Algorithmic Approach

Given the above definition of coverage $\mathcal{C}(p(\cdot), \omega(\cdot))$, our goal is an algorithm which will compute the coverage given the video stream produced by the camera trajectory $(p(\cdot), \omega(\cdot))$. We will use a deep learning pipeline for computing the coverage. Unlike the case of depth estimation, our pipeline will need to be supervised, as there is no straightforward unsupervised loss that can be used. Therefore, we need labelled training data; we now describe the data, following which we describe the general learning approach.

1) Training Data: To gather training data, the most natural way to proceed would be to have physicians label video segments according to their coverage scores, and to use these labels as ground truth. This is a standard approach to learning classification, detection, and segmentation models. However, there is a problem with using this approach in the case of coverage: it turns out that physicians have considerable difficulty in accurately estimating coverage scores. To illustrate these issues, we asked physicians to label synthetic video clips, and then we compared the physicians' labels with the ground truth.

More specifically, our videos are synthesized based on a colon simulator developed by 3D Systems [45]. The colon is represented by a fully texture-mapped mesh, which can then be rendered using standard rendering engines; we have chosen to use Blender [46]. Many different trajectories can be generated by taking a base trajectory and adding randomly chosen smooth curves in both the position and orientation of the camera; by doing so, we can generate many different full simulated colonoscopy procedures. Each full procedure is then cut into short segments of 10 seconds, or 300 frames. Example simulated images are shown in Figure 4.

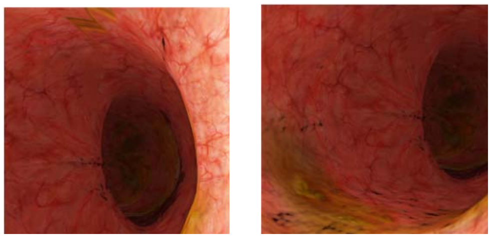
Fig. 4. Example simulated images.

The advantage of using such synthetic video clips is that we have the actual ground truth for such clips: given the geometric model of the colon (from which we render the clips), we can compute the coverage. Thus, the physicians' estimated coverage can easily be compared the ground truth. We generated 561 such videos, and asked six physicians to label them; these physicians were gastroenterologists, with experience levels between four and ten years in gastroenterology, with a median of seven years. We began by asking the physicians to label the actual coverage score, expressed as a percentage. Specifically, the task was explained to the physicians as "express the percentage of actual surface viewed out of the total possible surface that could be viewed"; the task was carefully explained to each physician, along with some training examples. The labels of the physicians, expressed as percentages, were then converted to fractions lying in [0, 1]. When compared with the ground truth labels, physicians had a mean absolute error (MAE) of 0.177, which is very large. The magnitude of this error is best viewed by examining the scatter plot of physicians' scores vs. ground truth labels, see the right subfigure of Figure 10: ideal performance would lie along the diagonal, but in practice the points are very far away from the diagonal.

We were interested to see whether physicians' performance on the labelling task was due to the fact that the label was expected to be a continuous variable (a percentage / fraction), and that such a task might be difficult or unnatural for many physicians. We therefore gave the physicians a much simpler task, namely to decide whether in a given segment the colon was (1) "mostly covered", (2) "partially covered", or (3) "mostly not covered". This 3-way classification task should be relatively straightforward. There remained the issue of how to map ground truth coverage scores, which lie in [0, 1] to these categories. We therefore computed the mapping that maximized the physicians' accuracy on the task, i.e. that correlated best with the physicians' labels. The result was equally convincing: on this 3-way classification task, physicians achieved an accuracy of 64.5%. In fact, even when the "partially covered" and "mostly not covered" classes were combined, so that the classification task was now a binary task, the accuracy only increased slightly, to 67.6%. These results show quite definitively that physicians have quite a difficult time estimating coverage.

We note that in general, the synthetic clips tend to be easier to label than real colonoscopy clips: the motion is smoother and slower, and there are fewer distracting artifacts (e.g. spraying of fluids). Thus, we would assume that the conclusions drawn based on the above statistics would apply at least as much to real colonoscopy clips, and perhaps to an even greater degree. The evidence provided above convinced us that we would need to train on synthetic videos, but to do so in such a way that we could then generalize to real videos. We describe the manner in which we did this next.

2) The Algorithm: Our approach to learning coverage is to break the training regime into two separate stages. We begin by noting that a special case of coverage can be computed when only a single frame is considered. In this case, the trajectory is just a single pose p, ω, so that coverage as defined in Equation (5) becomes

$$\mathcal{C}_{single}(p, \omega) = \frac{\mu[\mathcal{A}(p, \omega)]}{\mu[\mathcal{V}(p, p)]}$$

Note from Equations (3) and (4) that both $\mathcal{V}$ and $\mathcal{A}$ are well-defined for a single frame. In the first stage, then, we train a per-frame network, whose input is a single depth image, and whose output is the coverage for that frame $\mathcal{C}_{single}(p, \omega)$. In practice, we use a vector of outputs of several coverages, each computed with different viewing angle and look-ahead parameters, corresponding to Δ_0 and Δ_1 in Equation (3). In the second stage, we strip the final layer off of the per-frame network, exposing the penultimate layer which is then taken as a feature vector. We then train a per-segment coverage network by taking as input the collection of feature vectors, one for each frame in the segment; and the output is the segment coverage $\mathcal{C}(p(\cdot), \omega(\cdot))$. The structure of the two stage procedure is shown in Figure 5.

Why proceed with a two stage procedure, rather than a single stage? There are three primary reasons:

- **Allows for Simple Domain Adaptation:** We are training on synthetic videos, but the ultimate goal is for the networks to predict coverage on real videos. The concern is that the networks – given their large capacity – may learn to predict coverage based on some minor artifacts of the synthetic videos which do not then generalize to real videos. To deal with this, the initial per-frame network learns a feature representation based on a rather coarse representation of the 3D geometry, namely the visibility. Only this feature vector is then used in the final per-segment network. Due to the geometric coarseness of these features, a very simple domain adaptation scheme may be employed. In particular, we use the "frustratingly easy" technique of Sun *et al.* [47], which applies an affine transformation to the second last layer of the per-segment network. This affine transformation causes the mean and covariance of this layer's output, computed over the real segments, to be transformed to match the corresponding statistics for the synthetic segments. This very simple domain adaptation technique is all that is required to achieve high performance on real segments, as we show in Section V-B.
- **Less Training Data:** Synthesizing full videos is a rather heavy operation, as each frame must be rendered, and a video of 5 minutes consists of 9,000 frames. The natural approach, which learns coverage directly from video segments, would require many such segments to converge; and this would necessitate the rendering of a very large number of frames. Using the two stage approach mitigates this problem: a modest number of video segments, on the order of hundreds, will still consist of many frames. The per-frame network will therefore have a lot of data to learn from (hundreds of thousands of images); whereas the per-segment network will be learning a much easier task, and can therefore learn on a considerably smaller amount of data (hundreds of segments) using a network with much lower capacity.
- **Inference Speed:** A natural candidate for the architecture of a direct approach is a 3D CNN; this is the standard architecture that is used in action recognition, for example [48]. Unfortunately, such networks are quite heavy, and cannot generally run in real-time. Other approaches for spatio-temporal data include combined recurrent-convolutional architectures [48], [49]. Our proposal, by contrast, is a straightforward convolutional architecture, which is very clean and easy to train.

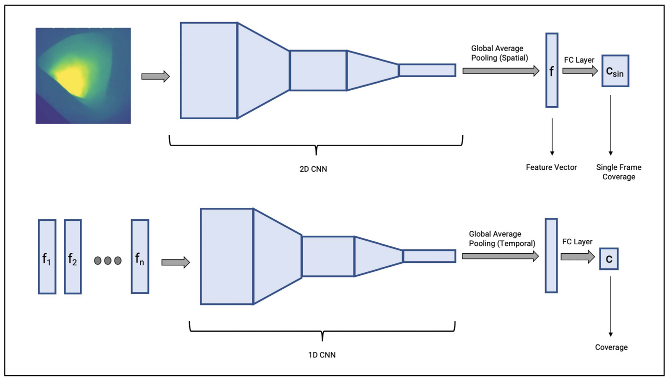

Fig. 5. Network structure. See accompanying description in the text.

It is natural to wonder whether one can combine the two stages into a single stage, via a unified loss function. We note that although this might be possible, there is a distinct advantage to keeping the training of the two stages separate, due to the fact that we have many more frames than we have segments. The per-frame model that is learned is therefore quite accurate, as it is trained on a very large number of examples. Once the per-frame model has been learned, the per-segment model can benefit from the per-frame model via the use of the representation that has been learned in the per-frame model. This enables a kind of transfer learning, which is very useful since there are far fewer segments than there are frames. If one were to learn on both frames and segments simultaneously, it is not clear if this transfer learning would work as well. For example, the part of the loss related to the segments might "drown out" the part of the loss related to the frames, which would be problematic, given the relatively small number of segments.

C. Network Architecture

We begin with the per-frame network. The input to the per-frame network is a depth image. We use a ResNet-50 architecture, strip off the final layer, and replace it with a fully connected layer which reduces to a vector of size three. Each entry of this vector is the visibility computed for different parameters (Δ_0, Δ_1). This is trained with an L_2 loss. See Figure 5 for an illustration of the per-frame network structure. After training is complete, we strip off the last layer of the per-frame network, so that the new output (previously, the penultimate layer) is a feature vector of length 2,048.

We now turn to the per-segment network. A segment is taken to be 10 seconds worth of video, which at 30 fps translates to 300 frames. Each frame is passed through the per-frame network, yielding a collection of 300 vectors, each of length 2,048. We consider this to be a 2-tensor, of length 300, and with 2,048 channels. This 2-tensor is then the input to the network, which is a 1D CNN. This network is relatively small, as we have a small number of training samples; there are six 1D convolutional layers, followed by average pooling over the temporal dimension and a final fully-connected layer (see Figure 5). The total number of parameters of the network is 20K, which is quite small.

In practice, run-time inference proceeds as follows. The current RGB frame is passed through the depth estimation network described in Section III. This depth image is then passed through the per-frame network, producing a vector of length 2,048. This vector is then appended to the vectors from the previous 299 frames (which were computed at previous time steps, and stored in memory) to create the 2D tensor which is then passed into the per-segment network. Regarding timing, we can compute the overall time required on a Titan Xp GPU using the numbers measured in [50], and scaled by the change in resolution (from 224×224 as used in [50] to 384×320). The depth network uses a ResNet-18 architecture, which requires 4.38 ms, while the per-frame network uses a ResNet-50 architecture which requires 12.49 ms. The per-segment network uses a non-standard architecture, which is nevertheless quite light; our own timing experiments indicate that it runs in 0.20 ms on a CPU, and therefore would require less on a GPU. The total time for all stages is therefore less than 17.07 ms, which would allow for frame-rates of up to 58 frames per second.

V. Results

A. Depth Estimation

1) Description of the Dataset: We have three different sources of data for evaluating our depth estimation algorithm. The first source is the dataset introduced in [19], which we refer to as the UCL dataset. This is a synthetic dataset, for which ground truth depth is available, consisting of 16,016 (RGB, depth) image pairs, with a train-test split of 10,556 vs. 5,460. The second source is based on synthetic videos we have generated, using the colon simulator developed by 3D Systems [45], which were then rendered using Blender [46]. Again, ground truth is available for this set, which consists of 187,369 (RGB, depth) image pairs, with a train-test split of 134,025 vs. 53,344. We refer to this dataset as the Google-Synthetic dataset. The final source is real de-identified colonoscopy videos from Orpheus Medical, which have been recorded at 16 mbps; we have acquired 3,049 such videos. These are very useful for training, given that our algorithm is unsupervised, but cannot be used for quantitative evaluation as no ground truth is available. We refer to this dataset as the Google-Real dataset.

2) Metrics: We report a few metrics for the quality of the depth estimation. The first metric is Mean Relative Error (MRE). In the ordinary way, MRE would be defined as

$$\text{MRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{d}_i - d_i|}{d_i} \tag{6}$$

where d_i is the ground truth depth of pixel i, $\hat{d}_i$ is the estimated depth, and the sum is over all the pixels in an image (or in multiple images). However, note the fact that the depth estimation algorithm is only correct up to scaling, so we must take this into account; furthermore, the above formula does not account for the case where the actual ground truth depth is 0. Thus, we emend the formula in (6) to read

$$\text{MRE} = \min_{\sigma}\frac{1}{n}\sum_{i=1}^{n}\frac{|\sigma\hat{d}_i - d_i|}{\max(d_i, \epsilon)} \tag{7}$$

The minimization allows us to choose the best scaling parameter for the test set, and the term in the denominator accounts for when the ground truth depth is 0.

Our hypothesis is that much of the error in the depth reconstruction comes near discontinuities. Specifically, if a discontinuity is correctly reconstructed, but its position is off by one pixel, then the MRE incurs a large loss. A measure closely related to the MRE is the Discontinuity Robust MRE, or DR-MRE:

$$\text{DR-MRE} = \min_{\sigma}\frac{1}{n}\sum_{i=1}^{n}\min_{j\in\mathcal{N}_i}\frac{|\sigma\hat{d}_i - d_j|}{\max(d_j, \epsilon)}$$

where $\mathcal{N}_i$ is the 3×3 neighbourhood around pixel i. If the position of the discontinuity is off by a single pixel, then the DR-MRE will be insensitive to this.

Finally, another way of dealing with the scaling issue is to note that order is preserved under scaling. Therefore, we may also verify whether the order of pairs of pixels is the same for both the ground truth and estimated depths. Given a pair of pixels, we define $r_{ij} = \mathbb{I}[d_i > d_j]$, where $\mathbb{I}[\cdot]$ is the indicator function; and similarly, $\hat{r}_{ij} = \mathbb{I}[\hat{d}_i > \hat{d}_j]$. Then we define the Depth Order Measure (DOM) to be

$$\text{DOM} = \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}}\left[r_{ij}\hat{r}_{ij} + (1 - r_{ij})(1 - \hat{r}_{ij})\right]$$

where the sum is taken over all pairs of pixels. In practice, the number of pairs is enormous, so a sampling strategy is employed. Note that the above measure is similar to the Rand Index.

TABLE I
PERFORMANCE OF THE DEPTH ESTIMATION ALGORITHM

	MRE	DR-MRE	DOM
Google-Synthetic	0.052	0.046	0.978
UCL [19]	0.168	0.079	0.933

3) Results: The performance of the depth estimation algorithm is reported in Table I. The numbers for the MRE indicate that on average, the estimated depth is within 5.2% or 16.8% of the true value for the Google-Synthetic and UCL datasets, respectively. However, examining the DR-MRE, we see that in the case of the UCL dataset, more than half of the error is due to small errors in discontinuity placement: the DR-MRE is only 7.9%. The gains for the Google-Synthetic dataset are considerably more modest in going from MRE to DR-MRE, indicating that the algorithm was better at placing depth discontinuities for this dataset. There are two potential reasons for this. First, Google-Synthetic is more than 10 times larger than UCL: the train set sizes are 134,025 vs. 10,556. Thus, there may have simply been enough data to learn the discontinuities better. Second, Google-Synthetic appears to be somewhat smoother than UCL.

We also note that Rau *et al.* [19] achieve MRE = 6.4% on the UCL dataset; this value is computed using the standard definition of MRE in Equation (6), rather than the scale insensitive version in (7). However, note that the algorithm of Rau *et al.* is fully supervised, whereas ours is completely unsupervised.

We now examine the DOM values. The DOM lies in [0, 1], and is quite high at 0.978 and 0.933 for the Google-Synthetic and UCL datasets, respectively. This indicates that order is preserved nearly all of the time. Our hypothesis is that such numbers would be sufficient to enable the coverage algorithm; this will be borne out in Section V-B.

We now turn to more qualitative results, by showing example depth maps from the various datasets; this is the natural way of judging the quality of the algorithm on the real dataset (given the lack of ground truth), but it also gives a better flavor of the performance on synthetic datasets. Synthetic results on the Google-Synthetic dataset are shown in Figure 6; note the striking similarities between the ground truth depth maps and their estimated counterparts. Real results on the Google-Real dataset are shown in Figure 7. One can see the overall features of the colon are captured nicely, including the depth of the "tunnel" down the lumen, as well as the folds.

B. Coverage

1) Description of the Dataset: The dataset comes from two sources. The first source consists of synthetic videos which were generated using the colon simulator developed by 3D Systems [45], and then rendered using Blender [46]. These videos are then divided into segments of 10 seconds duration, i.e. 300 frames; in total, 561 such video segments were generated. Each of these segments possesses a ground truth coverage label in $[0, 1]$. Note that in experiments, we use 5-fold cross-validation, allowing us to test on all

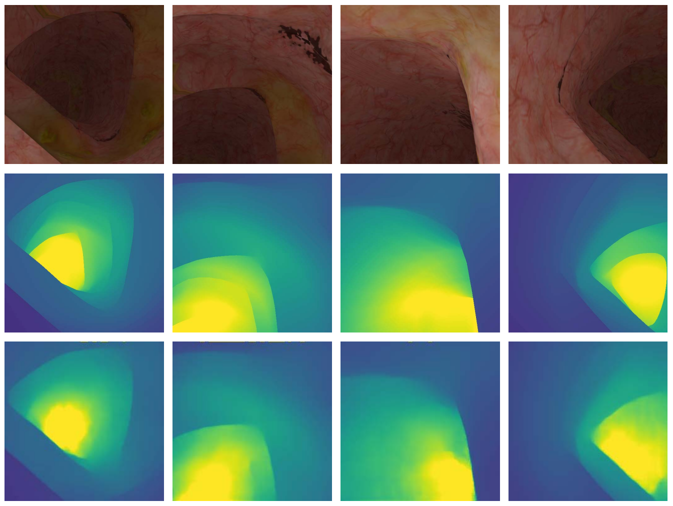
Fig. 6. Depth estimation on the Google-Synthetic dataset. Top: RGB image. Middle: ground truth depth map. Bottom: estimated depth map. Yellow is deeper, blue is more shallow.

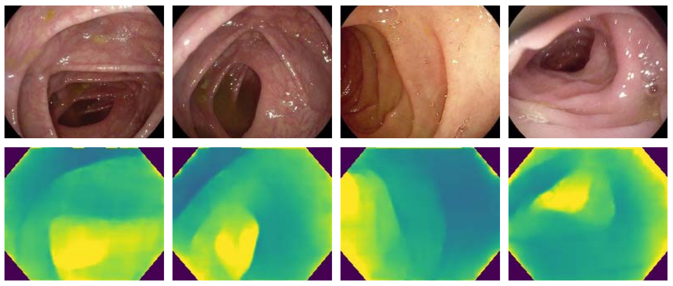
Fig. 7. Depth estimation on the Google-Real dataset. Top: RGB image. Bottom: estimated depth map. Yellow is deeper, blue is more shallow.

561 sequences. In addition, each such video was given a coverage value by a gastroenterologist, which allows us to compare C2D2's performance to that of human experts. This setup has already been discussed at length in Section IV-B. We are releasing this dataset, which is located at https://dl.google.com/datasets/CC20/Google-CC20-dataset.tar.gz.

The second source consists of real videos, which are full colonoscopy procedures which have been recorded at 16 mbps; this is the Google-Real dataset, which has already been described in the context of depth estimation. As in the case of the synthetic videos, the real videos are divided into segments, which are randomly chosen 10 seconds subsequences. These segments do not possess a ground truth label. As discussed at length in Section IV-B, human experts are not particularly accurate in estimating coverage – either in terms of the MAE of coverage, or accuracy on simpler classification tasks. Thus, one cannot use physicians to provide ground truth labels for real sequences. Instead, for quantitative validation on real segments we use an "expert verification" scheme, which we detail shortly, in the discussion concerning per-segment results.

2) Results: Per-Frame: We begin by describing results for the per-frame network. Although the per-frame task does not represent our final goal, it is nevertheless interesting to report results on this first stage of our two stage mechanism. We use

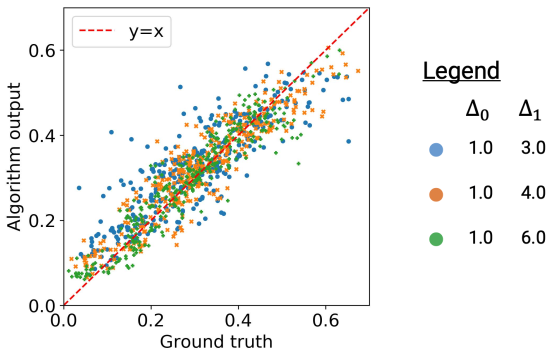

Fig. 8. Scatter plot of the predicted single frame coverage vs. true single frame coverage. The color denotes different parameter values for the visibility computation, see Section IV.

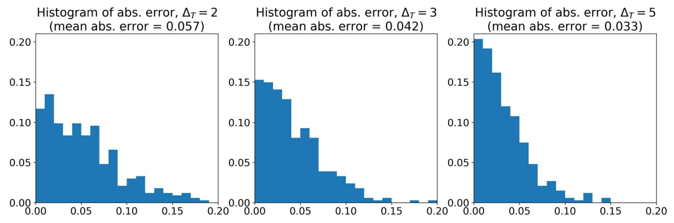

Fig. 9. Histograms of Mean Absolute Error (MAE) of the per-frame visibility network, for three separate parameter pairs (Δ_0, Δ_1). Left: (1.0, 3.0); middle: (1.0, 4.0); right: (1.0, 6.0).

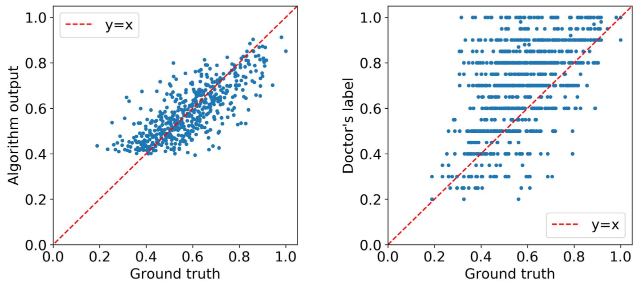

Fig. 10. Scatter plots of the predicted coverage vs. true coverage on synthetic sequences. Left: C2D2's performance. Right: physicians' performance. C2D2's performance is considerably better.

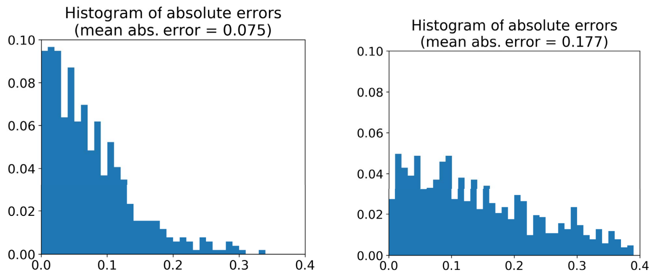

Fig. 11. Histograms of Mean Absolute Error (MAE) of coverage on synthetic sequences. Left: C2D2's performance, MAE = 0.075. Right: physicians' performance, MAE = 0.177. C2D2's performance is 2.4 times better.

three different values for the pair (Δ_0, Δ_1), namely (1.0, 3.0), (1.0, 4.0), and (1.0, 6.0). We first show a scatter plot of the predicted single frame coverage vs. true single frame coverage, for the test set in Figure 8; each color corresponds to a different value of (Δ_0, Δ_1) as shown in the legend. As can be seen, the network succeeds in learning, fairly well, how to predict single frame coverage.

A more quantitative measure of the performance is the Mean Absolute Error (MAE), which we report separately for each of the parameter values, shown in Figure 9. The MAEs range between 0.033 and 0.057, which is quite reasonable. It is crucial to remember that the actual performance is immaterial in the end; rather, this network is used as a feature extractor for the ultimate goal, which is per-segment coverage. Nonetheless, it is important that the feature extractor be informative, and achievement of the intermediate goal of single frame coverage prediction indicates that this is indeed the case.

3) Results: Per-Segment: We begin with a discussion of C2D2's performance on synthetic video segments, for which we have ground truth. As in the previous section, we show the scatter plots of the predicted coverage vs. true coverage. In Figure 10, the left plot shows C2D2's performance, while the right plot shows the physicians' performance. Ideal performance entails clustering on the diagonal; as can be seen, C2D2's performance is considerably better than that of the physicians.

To further quantify the difference in performance, we examine the MAE for both C2D2 and the physicians, see Figure 11. C2D2 attains MAE = 0.075, while the physicians receive MAE = 0.177. By this metric, C2D2's performance is 2.4 times better, clearly demonstrating the system's ability to outperform human experts.

We now turn to quantitative performance on real video segments. As noted above, physicians have difficulty with labelling coverage on synthetic sequences (which tend to be easier than real sequences), incurring an MAE of 0.177. Furthermore, as described in depth in Section IV-B, physicians also have difficulty with the simpler scenario of labelling sequences according to a 3-way classification task, in which the goal is to decide whether in a given segment the colon was (1) "mostly covered", (2) "partially covered", or (3) "mostly not covered". In particular, the physicians' accuracy on this task was 64.5%. Thus, we cannot simply have physicians label video clips according to these 3 classes and compare C2D2's predictions to these classes, as the physicians' labels are far too noisy.

Instead, we use a technique which is a variant of that used in the generative modelling literature [51]: we ask the physicians to judge the algorithm's result. More specifically, we present the physician with both the video segment, as well as C2D2's output, mapped to one of the three classes mentioned above; the physician is then asked whether they "agree" or "disagree" with C2D2's prediction. In order to enable this task, we must have a way of mapping C2D2's coverage score in [0, 1] to the three classes. We did this by examining a small number of clips – prior to the physicians' annotation – and deciding on a sensible set of bins by eyeballing. The bins were taken to be [0, 0.4), [0.4, 0.8), and [0.8, 1].

The results of this performance evaluation are shown in Table II. Two of the six physicians mentioned in

TABLE II
PERFORMANCE OF C2D2 ON REAL SEQUENCES

Total # Clips	# Clips with Physician Agreement with C2D2	Percent Physician Agreement with C2D2
301	280	93.0%

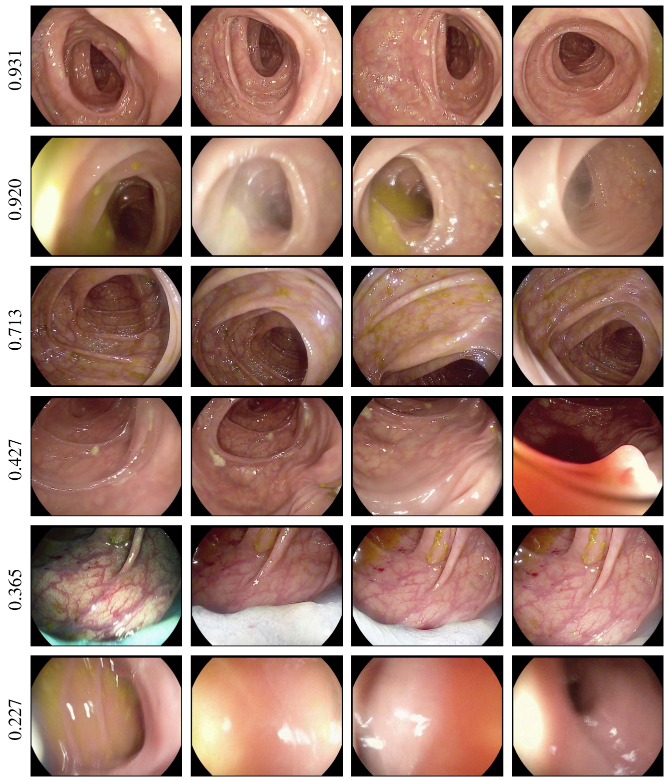

Fig. 12. Examples of C2D2's score on real sequences; in each case, we show four frames from the sequence, and we report C2D2's coverage score on the left of the row. The rows are arranged in order of descending coverage score. See accompanying description in the text.

Section IV-B (with 7 and 4 years experience as gastroenterologists) were given 385 clips; each clip was examined by a single physician. The physicians rejected 84 of the clips as not relevant for the coverage task – due to the presence of spraying of fluids, lack of motion due to a polyp being removed, etc. This left 301 clips, in which there was agreement on 280. Thus, in total there was physician agreement with C2D2's prediction on 93.0% of the clips, which speaks to the accuracy of the algorithm.

Finally, we turn to qualitative performance on real video segments; representative results are shown in Figure 12, with the rows arranged in order of descending C2D2 coverage score. The top two rows show examples of high coverage scores; C2D2 reports scores of 0.931 and 0.920. In each of these cases, it is clear that one can easily see the lumen, and the "tunnel" going down the center of the colon. The second row is interesting, in that the sequence is not as clean as that of the first row: the images are blurry, and there is also fecal material present. Nevertheless, C2D2 succeeds in reporting a high score. The third row shows a colon which is mostly covered, but it is clear that the sight line to the lumen is not straight on, and therefore various parts of the colon are missed to some degree (this is particularly noticeable in the third frame); C2D2's score of 0.713 is therefore quite plausible. The fourth row shows a partially covered colon: the bottom may be seen clearly, but the top is not visible. C2D2 accordingly gives a score of 0.427. The fifth row shows a somewhat similar example, except now more parts of the colon are clearly less visible: one cannot see the lumen at all, as compared to the fourth row, where part of it is somewhat visible. C2D2 assigns a lower score of 0.365 in this case. Finally, the sixth row shows an example in which much of the sequence is facing the intestinal wall, with occasional very partial views. Such a sequence sensibly receives a very low score of 0.227. In summary, C2D2 succeeds in returning coverage scores on real sequences which pass the eyeball test.

VI. CONCLUSIONS

We have presented C2D2, a new technique for computing coverage of a colonoscopy procedure, and we have demonstrated the accuracy of the technique on a large scale dataset. To the best of our knowledge, this is the first time a coverage algorithm has been evaluated on such a dataset. Our results show that C2D2 outperforms human experts by a wide margin on synthetic datasets with ground truth, and has a 93.0% agreement with physicians on real videos. Furthermore, as a building block used in achieving the goal of coverage, we have presented a depth estimation algorithm which is the first unsupervised, calibration-free method to be applied in the colonoscopy domain. This algorithm has also been shown to attain very promising results. In the future, we plan to test the efficacy of the coverage algorithm in a live clinical setting.

ACKNOWLEDGMENT

The authors would like to thank Gaddi Menahem and Yaron Frid of Orpheus Medical Ltd., for helping in the provision of data. They would also like to thank Ariel Ben Moshe, Eran Negrin, Smadar Magal, and Ran Bronstein of 3D Systems for their great help in describing the details of their colon simulation system. Finally, they would like to thank all of their team members and collaborators who worked on this project with them: David Ben Shimol, Nadav Rabani, Chen Barshai, Nia Stoykova, Jesse Lachter, and Ori Segol.

REFERENCES

[1] *Colorectal Cancer Fact Sheet 2018*. Accessed: Jan. 8, 2020. [Online]. Available: http://gco.iarc.fr/today/data/factsheets/cancers/10_8_9-Colorectum-fact%-sheet.pdf
[2] *Cancer Facts & Figures 2019*. Accessed: Nov. 26, 2019. [Online]. Available: https://www.cancer.org/research/cancer-facts-statistics/all-cancer-fact%s-figures/cancer-facts-figures-2019.html
[3] A. Leufkens, M. van Oijen, F. Vleggaar, and P. Siersema, "Factors influencing the miss rate of polyps in a back-to-back colonoscopy study," *Endoscopy*, vol. 44, no. 5, pp. 470–475, May 2012.
[4] Y. M. Lee and K. C. Huh, "Clinical and biological features of interval colorectal cancer," *Clin. Endoscopy*, vol. 50, no. 3, p. 254, 2017.
[5] M. F. Kaminski *et al.*, "Increased rate of adenoma detection associates with reduced risk of colorectal cancer and death," *Gastroenterology*, vol. 153, no. 1, pp. 98–105, Jul. 2017.

[6] G. Urban, P. Tripathi, T. Alkayali, M. Mittal, F. Jalali, W. Karnes, and P. Baldi, "Deep learning localizes and identifies polyps in real time with 96% accuracy in screening colonoscopy," *Gastroenterology*, vol. 155, no. 4, pp. 1069–1078, 2018.

[7] P. Wang *et al.*, "Real-time automatic detection system increases colonoscopic polyp and adenoma detection rates: A prospective randomised controlled study," *Gut*, vol. 68, no. 10, pp. 1813–1819, 2019.

[8] D. K. Rex, "Who is the best colonoscopist?" *Gastrointestinal Endoscopy*, vol. 65, no. 1, pp. 145–150, Jan. 2007.

[9] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2043–2050.

[10] M. Turan, Y. Almalioglu, H. Araujo, E. Konukoglu, and M. Sitti, "Deep EndoVO: A recurrent convolutional neural network (RCNN) based visual odometry approach for endoscopic capsule robots," *Neurocomputing*, vol. 275, pp. 1861–1870, Jan. 2018.

[11] R. Ma, R. Wang, S. Pizer, J. Rosenman, S. K. McGill, and J.-M. Frahm, "Real-time 3D reconstruction of colonoscopic surfaces for determining missing regions," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Berlin, Germany: Springer, 2019, pp. 573–582.

[12] L. Wu *et al.*, "Randomised controlled trial of WISENSE, a real-time quality improving system for monitoring blind spots during esophagogastroduodenoscopy," *Gut*, vol. 68, no. 12, pp. 2161–2169, 2019.

[13] D. Chen *et al.*, "Comparing blind spots of unsedated ultrafine, sedated, and unsedated conventional gastroscopy with and without artificial intelligence: A prospective, single-blind, 3-parallel-group, randomized, single-center trial," *Gastrointestinal Endoscopy*, vol. 91, no. 2, pp. 332–339, Feb. 2020.

[14] J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.

[15] M. Turan *et al.*, "Unsupervised odometry and depth learning for endoscopic capsule robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1801–1807.

[16] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1851–1858.

[17] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8977–8986.

[18] R. J. Chen, T. L. Bobrow, T. Athey, F. Mahmood, and N. J. Durr, "SLAM endoscopy enhanced by adversarial depth prediction," 2019, *arXiv:1907.00283*. [Online]. Available: http://arxiv.org/abs/1907.00283

[19] A. Rau *et al.*, "Implicit domain adaptation with conditional generative adversarial networks for depth prediction in endoscopy," *Int. J. Comput. Assist. Radiol. Surgery*, vol. 14, no. 7, pp. 1167–1176, Jul. 2019.

[20] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, Dec. 2016.

[21] A. R. Widya, Y. Monno, M. Okutomi, S. Suzuki, T. Gotoda, and K. Miki, "Whole stomach 3D reconstruction and frame localization from monocular endoscope video," *IEEE J. Transl. Eng. Health Med.*, vol. 7, pp. 1–10, 2019.

[22] C. Wu, "Towards linear-time incremental structure from motion," in *Proc. Int. Conf. 3D Vis.*, Jun. 2013, pp. 127–134.

[23] M. A. Armin, G. Chetty, H. De Visser, C. Dumas, F. Grimpen, and O. Salvado, "Automated visibility map of the internal colon surface from colonoscopy video," *Int. J. Comput. Assist. Radiol. Surgery*, vol. 11, no. 9, pp. 1599–1610, Sep. 2016.

[24] Q. Zhao, T. Price, S. Pizer, M. Niethammer, R. Alterovitz, and J. Rosenman, "The endoscopogram: A 3D model reconstructed from endoscopic video frames," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Berlin, Germany: Springer, 2016, pp. 439–447.

[25] D. Hong, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "3D reconstruction of virtual colon structures from colonoscopy images," *Computerized Med. Imag. Graph.*, vol. 38, no. 1, pp. 22–33, Jan. 2014.

[26] G. Pinheiro, P. Coelho, M. Salgado, H. P. Oliveira, and A. Cunha, "Deep homography based localization on videos of endoscopic capsules," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2018, pp. 724–727.

[27] M. Wang, Q. Shi, S. Song, C. Hu, and M. Q.-H. Meng, "A novel relative position estimation method for capsule robot moving in gastrointestinal tract," *Sensors*, vol. 19, no. 12, p. 2746, Jun. 2019.

[28] M. Aghanouri, A. Ghaffari, N. D. Serej, H. Rabbani, and P. Adibi, "New image-guided method for localisation of an active capsule endoscope in the stomach," *IET Image Process.*, vol. 13, no. 12, pp. 2321–2327, Oct. 2019.

[29] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2014, pp. 834–849.

[30] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[31] R. Wang, M. Schworer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3903–3911.

[32] Y. Almalioglu, M. R. U. Saputra, P. P. B. D. Gusmao, A. Markham, and N. Trigoni, "GANVO: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5474–5480.

[33] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 817–833.

[34] H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 822–838.

[35] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular visual odometry through unsupervised deep learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7286–7291.

[36] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1983–1992.

[37] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 340–349.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[39] N. Mahmoud, T. Collins, A. Hostettler, L. Soler, C. Doignon, and J. M. M. Montiel, "Live tracking and dense reconstruction for handheld monocular endoscopy," *IEEE Trans. Med. Imag.*, vol. 38, no. 1, pp. 79–89, Jan. 2019.

[40] F. Mahmood, R. Chen, and N. J. Durr, "Unsupervised reverse domain adaptation for synthetic medical images via adversarial training," *IEEE Trans. Med. Imag.*, vol. 37, no. 12, pp. 2572–2581, Dec. 2018.

[41] L. Maier-Hein *et al.*, "Comparative validation of single-shot optical techniques for laparoscopic 3-D surface reconstruction," *IEEE Trans. Med. Imag.*, vol. 33, no. 10, pp. 1913–1930, Oct. 2014.

[42] R. Garg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Proc. ECCV*, 2016, pp. 740–756.

[43] B. Ummenhofer *et al.*, "DeMoN: Depth and motion network for learning monocular stereo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5038–5047.

[44] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5667–5675.

[45] *3D Systems GI Mentor Platform*. Accessed: Dec. 23, 2019. [Online]. Available: https://simbionix.com/simulators/gi-mentor/gi-mentor/

[46] *Blender 3D Pipeline*. Accessed: Dec. 23, 2019. [Online]. Available: https://www.blender.org/

[47] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2058–2065.

[48] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6299–6308.

[49] M. Zhu and M. Liu, "Mobile video object detection with temporally-aware feature maps," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5686–5695.

[50] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[51] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.