Reducing the Pain: A Novel Tool for Efficient Ground-Truth Labelling in Images

Christopher J. Rapson, Boon-Chong Seet, M. Asif Naeem, Jeong Eun Lee, Mahmoud Al-Sarayreh, Reinhard Klette
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland 1010, New Zealand

***Abstract*—Machine Learning solves more image processing problems every year, yet it is still reliant on painstaking manual ground truth labelling. Segmentation labels require higher accuracy and more clicks than bounding boxes or classification labels. To accelerate the labelling task, a More Efficient Labelling Tool (MELT) has been developed which incorporates features from existing tools and adds some novel ones. The new features are automatic zoom to existing bounding boxes and tracking of arbitrarily shaped objects. Zooming to bounding boxes makes it easy to upgrade bounding box labels to segmentation masks, or to label parts of an object, such as lights on a vehicle. Tracking is available in other tools for rectangular objects such as bounding boxes, but many objects including vehicle lights are not rectangular. The user is given the freedom to create labels with a brush, polygon or superpixel, with customisable label names and colours.**

Using MELT, a dataset of over 800 images has been prepared with image segmentation labels for vehicle head lights and tail lights. Labels are provided for download as mask files. As there is currently no comparable dataset available, it is hoped that this will become a benchmark for researchers working on detecting and tracking vehicle lights.

***Index Terms*—image labelling, annotations, segmentation, vehicle lights, image dataset**

1. Introduction

Machine Learning (ML) has revolutionised the field of image and vision computing. The most expensive and time consuming phase of any ML project is labelling the *Ground Truth* (GT) in a large dataset of images. The labelling task becomes sequentially more difficult when moving from classification, to detection, to segmentation. Segmentation is most difficult, but also most powerful, allowing ML to address a wider range of tasks, and more closely imitate natural vision. Even an established expert in the object detection field notes that "I'm probably a true believer in masks" [1], since segmentation masks more closely describe reality than a *bounding box* (bbox). Large projects such as Cityscapes are today working at the level of semantic segmentation, see for example [2].

Using the right software when labelling images has enormous potential to save time, effort and money. Despite this potential, many of the available software tools lack either a user-friendly interface or adequate features to enable the user to label GT for image segmentation quickly and accurately. This paper presents a More Efficient Labelling Tool (MELT) which aims to make it as easy as possible for a user to create image segmentation GT labels. It is compared to other tools, and shown to provide more features than any of the available options, along with a comfortable user interface.

MELT has been developed as part of a project to automatically detect brake lights and head lights on vehicles [3]. MELT's features will be demonstrated on this dataset, and the dataset will be made available for other researchers who may be working on automatic detection of vehicle lights. To our knowledge, there is no other publicly available dataset of images where vehicle lights GT has been labelled. Some researchers may want to detect brake lights to anticipate deceleration of the vehicle in front. Others may want to track headlights for adaptive shaping of high-beams. Our own project is looking at using the lights for a form of wireless communication network using visible light and cameras [4]. Since transmitters (LEDs) and receivers (cameras) are already installed in many new vehicles, this could be a low-cost complement to conventional radio frequency (RF) wireless communication.

In the remainder of this paper, Section 2 gives an overview of existing tools for segmentation GT labelling, Section 3 describes features of the new tool, and Section 4 describes the dataset which has been labelled using MELT.

2. Review of Existing Tools

An extremely large number of tools already exist for labelling images. This section will describe the features and limitations of a selection of those that can be used for image segmentation. All tools which can be used to segment an image can trivially be adapted to label bboxes for object detection.

LabelMe [5] is probably the most well known online tool. It has a large user base, and is maintained to a high standard. Since it is accessed online, it is inherently independent of a user's operating system, and doesn't require any software to be installed. Because images stored in *LabelMe*'s database are publicly accessible, it may be possible to outsource the labelling work. Some users will provide labels 'for fun', others can be incentivised by payment. For example, *Mechanical Turk* [6] allows researchers to reward contributors with a moderate fee, and provides an interface to tools such as LabelMe. In general, the results of outsourced labelling should be checked to confirm that the masks are positioned to an adequate

precision, and given accurate and consistent labels. Some masks are very precise, and some are rough polygons with less than 10 vertices for a vehicle. A quick browse of *LabelMe* shows a wide variety of labels which may or may not be synonymous for a given purpose, e.g. car, Car, car side, car crop, car occluded, van. Occasionally, labels are incorrect, such as bmw_car for a vehicle which is actually a Mercedes. The preprocessing which would be necessary to confirm the accuracy and consistency of *LabelMe* data is nearly as manually intensive as labelling.

LabelMe has the basic capability to zoom and scroll to closely examine different parts of the image. If other labels are obstructing the view, there is an option to hide or reveal all existing object labels. Objects can be labelled for segmentation with polygons, or by nominating certain pixels as foreground and others as background. An in-built algorithm then infers the boundary of the object. The in-built algorithm is usually more accurate than polygons, but it may take several iterations before the definitions of foreground and background give the desired result.

Drawbacks of the *LabelMe* interface include the lack of customisation for label colours and the lack of standardisation for labels as already mentioned. For this author's personal preference, the broad lines (approximately 10 pixels) are too bulky, and tend to discourage precise labelling. It is difficult to edit existing object labels, as new vertices can not be added to an existing polygon. Labels created by another user cannot be edited at all. The online nature of *LabelMe* means that even with a fast internet connection, there is still some noticeable delay. Perhaps the most important motivation for not using *LabelMe* was the complex format in which the labels are stored. Labels are stored in folders called `Masks`, `Scribbles`, and `Annotations`. The first two contain one `PNG` file for each object, i.e. dozens of files per image. Annotations are verbose `XML` files.

Labelling apps distributed with Matlab are another popular option. Matlab runs on all common operating systems, but is proprietary. License costs may be prohibitive. The *Image Labeler App* has many of the same basic features as *LabelMe* e.g. zoom, pan, define custom labels. Zooming is better, as the user has the choice of the scroll wheel, single click, or selecting an area to zoom into. The App allows objects to be labelled as polygons, smart polygons, with a brush or with a flood tool. When editing object labels, any combination of tools can be used together. The "smart polygon" was relatively disappointing in this author's testing, until the user takes advantage of the "smart polygon editor". The initial smart polygon is usually not a good match to the object boundary, however this can be corrected by nominating pixels as foreground and background, iterating as for *LabelMe*. Another promising feature is the interface for automated object detection, which can use conventional image processing algorithms or algorithms trained by ML. Unfortunately, the performance of the provided example was disappointing - the person detector failed to detect a person which was the sole 'object' in test images. Even by tuning the settings, it was not possible to have the correct detections outnumber the false positives. Creating a custom automation routine is complex. Despite the difficulties, current trends suggest that this feature will be widely used in the future, especially ML algorithms which suggest GT image labels, so that the user effort is reduced to accepting, rejecting or making minor edits.

The *Automated Driving Toolbox* provides the *Ground Truth Labeler App*, which has the additional feature of tracking bboxes in video. The algorithm uses feature detection, *Kanade-Lucas-Tomasi* (KLT) tracking and a matrix transformation to find the location of the object's bbox in successive frames. Custom tracking algorithms are allowed, although the procedure to create them is again relatively complex. Tracking is only available for bounding boxes, not labels created with the polygon, brush or flood tools.

RectLabel [7] is a proprietary annotation program for mac only. It has a slick interface, allowing labelling with bboxes, polygons and cubic beziers. The annotation dialog is advanced, providing automated classification suggestions and allowing many attributes of an object to be labelled e.g. colour, brand, view angle. Objects can be searched and quickly zoomed in on. *RectLabel* supports shortcut keys which speed up the labelling process.

Some other notable tools include one developed by *Vicomtech* [8] which allows labelling by superpixels [9]. Superpixels partition the image into groups of neighbouring pixels with similar colour, providing quick and easy separation of many objects in a scene. Subject to certain conditions, superpixel boundaries are generally at the pixels with strong colour gradients, which usually gives a more accurate label than a human can achieve by setting polygon vertices. Superpixels are comparable to "smart polygon" approaches, with the distinct advantage that they require many fewer clicks. Superpixels and smart polygons struggle with objects that have low contrast compared to their surrounding pixels. For example, tail lights on red vehicles or head lights on white vehicles are usually not well identified. Conversely, superpixels and smart polygons may give misleading results if there are artefacts from ambient light or shadow in an image. For example, where an object (such as a tail light) is curved rather than planar, part of the object may be in direct sunlight while the rest is in shadow. Superpixel and smart polygon algorithms will predict a boundary where the lighting gradient is strongest, rather than at the true edge of the object. *PolyRNN* [10] uses a different "smart polygon" approach. Instead of nominating foreground and background pixels, the user can directly manipulate the boundary produced by the algorithm. The effect of each user input is easier to predict, which reduces the number of iterations necessary to accurately define the object outline. *VIA* [11] provides tracking of a face's bbox using *Faster RCNN* [12].

3. A More Efficient Labelling Tool (MELT)

The following section describes a tool which incorporates many of the features described above, and a few novel features. The tool has been developed using the *Matlab App Designer* interface. It can be packaged with the *Matlab Compiler Runtime* (MCR) for execution without a Matlab license on any Operating System. Consider first the graphical interface, shown in Fig. 1. In the top left are buttons to "zoom" and "pan" to the region of interest within the figure. Zoom functionality provided by *Matlab* allows the use of the scroll wheel, single click or selecting

a region of interest. The "select" button deactivates the zoom and pan functionality. Tick boxes allow the label mask and/or the image to be hidden. It can be useful to hide the mask since overlaying the label's colour can obscure details of the image. Likewise it can be difficult to be sure that a yellow label on a yellow background is positioned correctly, so temporarily hiding the image can help in that case.

Going down the right-hand panel, the folder button allows the user to choose a folder to load images from. The drop-down list offers different methods of labelling the image. Each method will be explained in detail in Section 3.1. In order to edit existing labels or correct mistakes, selected pixels are reset to background when the "Erase" button is activated. The next two fields set parameters for the labelling methods. In the example shown, "Brush size" is the only relevant option, and "SP Compactness" is greyed out.

The list of labels shows their customisable names and colours. Reddish colours have been chosen to represent tail lights and yellowish colours for head lights. The list is scrollable in case it is long or the window is small. Buttons below the list allow for adding a new label (+), editing, or removing (-) the selected label. Sets of labels can be saved as a *.mat* file and loaded later, or by a collaborator.

When the user is satisfied with their labelling of an image, they can press the large "$>$'' or "$<$" button to move to the next or previous image, respectively. The mask will be saved to the file shown in the bottom right corner of the window, in the same folder as the image. The user may also move to the next or previous image without saving the mask. All annotations will be lost, which is an easy way of starting over if necessary. For navigating large folders, fast-forward ($>>$) and rewind ($<<$) buttons skip 10% of the files in the folder with one click. There are also buttons to skip to the first ($|<$) or last ($>|$) image in the folder.

The last two checkboxes labelled "Track Objects and "Bboxes only" activate advanced features which will be described in Sections 3.2 and 3.3. The remainder of the window is taken up by the image, with its filename shown above it.

3.1. Labelling Methods

The simplest labelling method is the "brush". Clicking in the image will mask an $n \times n$ square of pixels with the currently active label. The brush tool is most useful for small masks, or small edits to larger masks, on the order of a few pixels. The size of the square is set by the "Brush Size" option. Figure 2 shows a label with a 5×5 brush for the right head light, and an almost completed polygon for the left head light.

The polygon tool allows more complex shapes to be labelled with a small number of clicks. Left clicks add vertices to the polygon, right clicks cancel the polygon in progress. Figure 2 shows a completed polygon and a polygon in progress. Clicking on the first point (magenta) closes the polygon, and masks all pixels within the polygon as the active label. Unlike other implementations, the polygon vertices are not saved. This allows the mask pixels to be edited by any of the labelling methods. For example, the brush can be used to precisely add or erase single pixels which the polygon may have missed.

Superpixels can be calculated quickly and plotted over the image. If the superpixels do not initially align well with the object boundaries, they can be adjusted by three parameters. Firstly, the dropdown box allows a choice between *SNIC*, *SLIC* and *SLIC0* algorithm variants. *SNIC* (Simple Non-Iterative Clustering) is a state-of-the art superpixel algorithm [14], while *SLIC* (Simple Linear Iterative Clustering) and *SLIC0* have been included with Matlab's image processing toolbox for some time. SLIC0 allows the algorithm to iteratively refine the compactness, while *SLIC* keeps this value constant. Compactness is the ratio of superpixel area to perimeter length. Lower numbers allow more flexibility in matching the shape to the pixel properties, while for larger numbers the superpixels become more square. The number of superpixels should be set roughly inverse to the size of the objects the user wants to label. A high number will result in many small superpixels, which means more clicking and can obscure the image. Figure 3 shows an example of a bright red tail light which can been labelled in just one click with the superpixel tool, despite its relatively complex shape. Other objects such as the hubcaps on the foreground vehicle, and the right tail light and license plate on the background vehicle have also been clearly segregated. The second image in Fig. 3 shows how the license plate of the foreground vehicle can be segregated by adjusting the superpixel parameters. Increasing the compactness makes the superpixel edges smoother, and decreasing the number of superpixels increases their size. In this case, the left tail light is only a few pixels wide and does not have significant contrast to the background, so the user should switch to brush or polygon to create that label.

The examples given not only demonstrate the features of the various labelling methods, they also show why it is important for a tool to provide the user with several options. Superpixels are fastest and most accurate for objects with high contrast that are neither too large nor too small. If superpixels can not be used, then the brush is best for small objects, and the polygon is best for large objects. Given that each method has its respective advantages and disadvantages, the user must be able to choose the most appropriate one for the task at hand.

3.2. Autozoom to Bounding Boxes

High level image classification and object detection are relatively mature. Several projects, including ours, are looking to the next step of labelling parts of images for example lights on vehicles, eyes in faces or clothes on a person. It can be very helpful to automatically zoom to the object of interest in order to label its parts more accurately. The checkbox "Bboxes only" provides this functionality. Compare Figs. 1 and 2 for an example of an image which is automatically zoomed to the bounding box of one vehicle. Bbox co-ordinates are read from a range of popular formats: KITTI, Berkeley Drive, Cityscapes, YOLO and SYSU. The script can be customised to only consider objects from relevant classes (in this case vehicles) and ignore the rest (such as buildings). It would be possible in the future to run a fast object detection routine like YOLO [1] to automatically generate bboxes even for

Figure 1. Graphical User Interface for the new tool. Red numbers indicate regions for 1) select, zoom, pan 2) show/hide the image and mask 3) select image folder 4) selection of labelling method and parameters 5) define labels and their properties 6) output file name 7) navigation between images 8) tracking objects and autozoom to bboxes. Image from the *Wilddash* dataset [13].

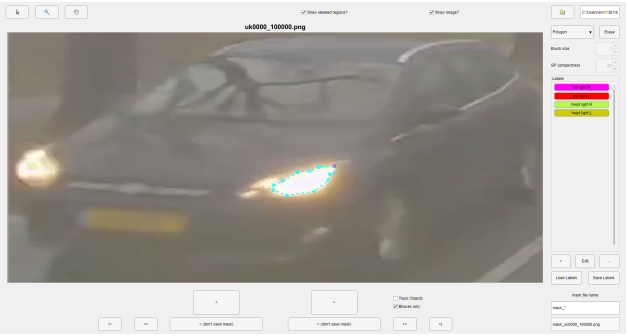

Figure 2. Examples of masks created with the brush and polygon tools. The figure is automatically zoomed to show the bounding box of a single vehicle. Image from the *Wilddash* dataset [13].

images where bbox labels have not been manually added. The autozoom feature accelerated the labelling task by a very significant amount.

3.3. Track Objects

In videos, objects' positions generally do not change significantly from one frame to the next. Small movements can easily be tracked. Labelling tools can make use of this insight to transfer manually created labels from one frame to a whole sequence of frames. As noted earlier, several tools allow bboxes to be tracked, but none were useful when it came to tracking objects for segmentation masks.

In order to track arbitrarily shaped objects, the labelled pixels in the mask are first grouped into "connected components". The convex hull of each connected component gives the vertices of a polygon that defines the object's boundary. Features (e.g. minimum eigenvalue corners) are

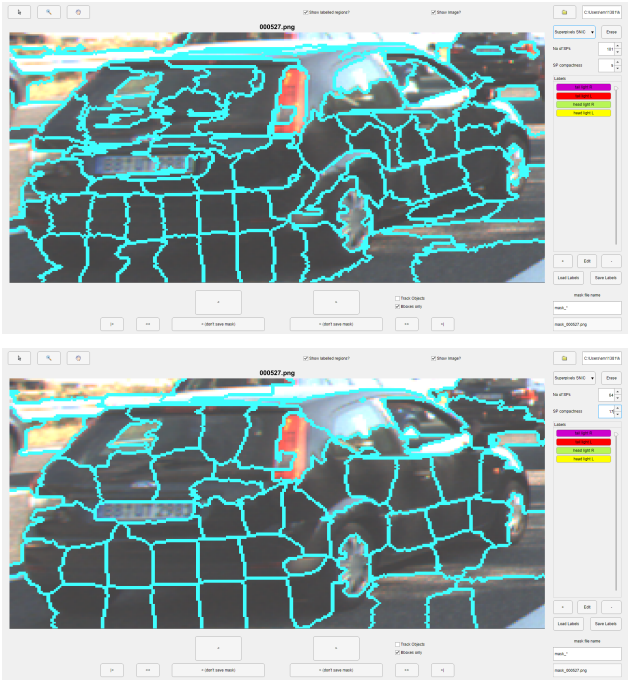

Figure 3. Examples of how superpixels can be used to quickly label regions with high contrast. In the first screenshot, the number of superpixels is set to 101, while the compactness is 5. In the second screenshot, the numbers are 64 and 18, respectively. The figures are automatically zoomed to show the bounding box of a single vehicle. Image from the *KITTI* dataset [15].

identified within the object's boundary. Since it can be difficult to identify enough features within small objects, and since vehicle lights tend to move with their immediate background, objects with less than 100 pixels are enlarged

by a factor of 4. Then matching features are found in the next image, using the KLT algorithm provided by *Matlab's Point Tracker*. Given the previous and updated location of a list of features, the program calculates the matrix operation which performs the required transformation, in a minimum squared error sense. Applying the matrix operation to the vertices of the polygon gives an updated polygon. The updated polygon can be converted into pixels, and those pixels are masked with the same label as before.

The routine tracks all labelled objects within an image, and can be performed forwards or backwards. Tracking dramatically reduces the time and effort required to annotate objects in a video.

The main limitation of this method is that the true motion is continuous, and discretising as pixels gives an inevitable quantisation error. The error is only a single pixel per frame, but over several frames, these errors accumulate into a drift. The same effect applies to KLT tracking of bounding boxes by other tools, however the errors are far more visible for segmentation masks. The effects are also more visible for smaller objects, where single pixels can be a significant fraction of their size. Another important limitation is that KLT tracking assumes no occlusion, and no change of shape. Change of shape is not a big problem for tracking vehicle parts, but occlusion is unfortunately quite common. The final limitation is that convex hulls assume convex shapes, so only filled polygons can be tracked. Holes will be filled in. It may be possible to compensate for these limitations by combining the estimated tracked position with minor corrections based on superpixels.

The advantages and limitations of KLT tracking can be observed in Fig 4. Without any manual interaction, the labels are propagated over 3 successive images. In the first image, most labels require no manual interaction at all. We observe that the right tail light of the second vehicle is no longer occluded, and its label should be added. The right tail light of the first vehicle is now outside the field of view, and is no longer tracked. The hole in the left tail light of the second vehicle has been filled in as a convex hull, and this label should be edited. In subsequent images, the drift becomes visible, and errors in the position of all labels should be corrected. KLT tracking can not replace manual labelling, but at least some of the user's work has been started for them. It should be noted that this video was recorded at 10 *frames per second* (fps). Using a faster frame rate such as 30 fps would improve the tracking accuracy.

In future, MELT could be improved by adding keyboard shortcuts. Anything that can be done with the keyboard increases productivity. As *Matlab's AppDesigner* matures, more features will be added such as hover text to explain the function of each button. MELT is available (by request to the corresponding author) for other researchers to use.

Figure 4. Labels for head and tail lights are manually created in the first image, and then automatically tracked over a sequence of images. Images from the KITTI dataset [15].

4. Dataset of Vehicle Lights

The main motivation and application for the new image labelling tool was to create a dataset of vehicle lights. This dataset will now be made available for other researchers. To our knowledge, it would be the first publicly available dataset of its kind. Researchers may want to use it to improve detection of brake lights in order to predict the future velocity of a vehicle in front. Or they may track head lights to detect oncoming vehicles at night. The original motivation for compiling the dataset is to detect and track pairs of vehicle lights for optical camera communication [3]. For this reason, neither the central brake light in cars nor the single head and tail lights on motorbikes are labelled. The lights on some distant vehicles have an extent of only a few pixels, which is comparable to blurring from motion, imperfect focussing or atmospheric distortion. These lights do not add useful information, so they have mostly not been labelled. See, for example, the vehicles beyond the intersection in Fig. 4.

TABLE 1. MASK FILES ARE GRAYSCALE PNG IMAGES, WHERE THE PIXEL VALUES SHOULD BE INTERPRETED AS FOLLOWS:

pixel value	label
0	background
1	head light Left
2	head light Right
3	brake light Left
4	brake light Right

The dataset includes the masks only, with links to the original images. Images were sampled from several vehicle image datasets from Asia, Europe and North America: *Berkeley Drive* [16], *Cityscapes* [2], *Foggy Driving* [17], *KITTI* [15], *SYSU* [18] and *Wilddash* [13]. Conditions range from bright sunlight to fog, rain, twilight and night. The mask files are grayscale PNG images, where the pixel values correspond to the label index, as shown in Table 1. In the future, it may be extended to more images, and video labels may also be released. Another obvious extension would be to label left and right indicator lights. It is hoped that researchers who make use of this dataset will use it as a benchmark, and publish their results to compare with others.

The dataset can be accessed from cerv.aut.ac.nz/vehicle-lights-dataset.

5. Conclusions

A More Efficient Labelling Tool (MELT) has been developed for image segmentation GT labelling. To address challenges posed by different objects in different lighting conditions, this tool allows the user the freedom to label their objects with a brush, polygon or superpixels. Superpixels are fastest for moderately sized objects that have high contrast to their surroundings. Polygons are useful for large or low contrast objects, and the brush is ideal for small details. To accelerate the manually intensive labelling task, the tool can automatically zoom to existing bbox labels, and track arbitrarily shaped objects in sequential frames of a video. These features have been packaged in a comfortable user interface.

The tool has been used to label vehicle head lights and tail lights. The GT labels have been provided as mask files in a publicly available dataset for other researchers to use.

Both the image labelling tool and the image dataset will continue to be developed in the future. New features could include a form of smart polygon, improved tracking or ML-based mask suggestions. Keyboard shortcuts will be another boost to productivity. External contributions to the dataset, for instance labels for indicators, central brake lights or other vehicle parts, would be welcomed.

References

[1] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," University of Washington, Tech. Rep., Apr. 2018. [Online]. Available: https://arxiv.org/pdf/1804.02767.pdf

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] C. J. Rapson, B.-C. Seet, P. H. J. Chong, and R. Klette, "Applying NOMA to undersampled optical camera communication for vehicular communication," in *2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU) (ICMU2018)*, Auckland, New Zealand, Oct. 2018.

[4] T. Nguyen, A. Islam, T. Hossan, and Y. M. Jang, "Current status and performance analysis of optical camera communication technologies for 5G networks," *IEEE Access*, vol. 5, pp. 4574–4594, 2017.

[5] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman, "LabelMe: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2007.

[6] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011, pMID: 26162106. [Online]. Available: https://doi.org/10.1177/1745691610393980

[7] R. Kawamura, "Rectlabel," Online, 2018. [Online]. Available: https://rectlabel.com

[8] Vicomtech, "Pixelwise annotator for ground truth generation," Online, Apr. 2017. [Online]. Available: https://www.youtube.com/watch?v=xBUT4nJDh20

[9] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 10–. [Online]. Available: https://ieeexplore.ieee.org/document/1238308/

[10] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with Polygon-RNN++," in *CVPR*, 2018.

[11] A. Dutta, A. Gupta, and A. Zissermann, "VGG image annotator (VIA)," http://www.robots.ox.ac.uk/ vgg/software/via/, 2016.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: https://ieeexplore.ieee.org/document/7485869/

[13] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "How good is my test data? Introducing safety analysis for computer vision," *International Journal of Computer Vision*, vol. 125, no. 1, pp. 95–109, Dec 2017. [Online]. Available: https://doi.org/10.1007/s11263-017-1020-z

[14] R. Achanta and S. Süsstrunk, "Superpixels and polygons using simple non-iterative clustering," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4895–4904.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *CVPR*, 2017.

[17] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018.

[18] L. Chen, "Sun Yat-sen University detection and tracking of moving objects (DATMO) dataset," Online, 2015. [Online]. Available: http://www.carlib.net/?page_id=35