Supplementary Material for CHI'21 Paper
Creating Accessibility Metadata for Mobile Applications from Pixels

1 ANNOTATION MATCH HEURISTICS FOR DATASET COMPOSITION ANALYSIS

This section presents a detailed description of the heuristics briefly summarized in Paper Section 3.3. These heuristics aim to match annotations with non-fullscreen UI elements exposed to the accessibility services. As shown in Fig. 1, we first categorized annotations and UI elements in each screen by on whether any bounding box B from a UI element contains, partially overlaps, or has no overlap with each annotation's bounding box A. We define "containment" as having at least 85% of A's area lies within B, and "overlap" as whether the intersection over union (IoU) of A and B is at least 5%. When a UI element contains only one annotation, we consider them as a match with each other. If an annotation has no overlap with any UI element, we consider them to not have a match with the exception of cases outlined in Appendix Section 1.4. The remaining sections outline our heuristics for the remaining possibilities: a UI element overlaps but does not contain an annotation, or it contains more than one annotation.

1.1 Looser Bounding Box Alignment

Some annotations do not line up exactly to the underlying UI element. For instance, annotators sometimes draw the horizontal aspect of bounding boxes more tightly than the underlying UI element's and the vertical aspect more loosely, or vice versa. Sometimes the underlying UI element's bounding box, which we extract from its exposed metadata, does not match its visual dimensions, at times cropping large portions in one direction and extending further in the other.

Based on our analysis of 150 screens, we defined the following heuristics to handle these cases. If an annotation has a relatively high IoU of at least 80% with a UI element, it is considered a match. As Pictures, Icons, and Text had the most inconsistent bounding boxes, any such annotations contained by a UI element is considered a match if it meets a looser threshold of 70%. For Text elements, multiple smaller UI elements may represent all of the text bounded by one annotation. To handle this situation, we applied a "reverse" containment algorithm for this case: any Text annotations that contain at least one UI element by this looser threshold also has a match. Annotations for short and wide annotation types like Page Controls match with UI elements that could be horizontally cropped if the element does not overlap with any other UI element.

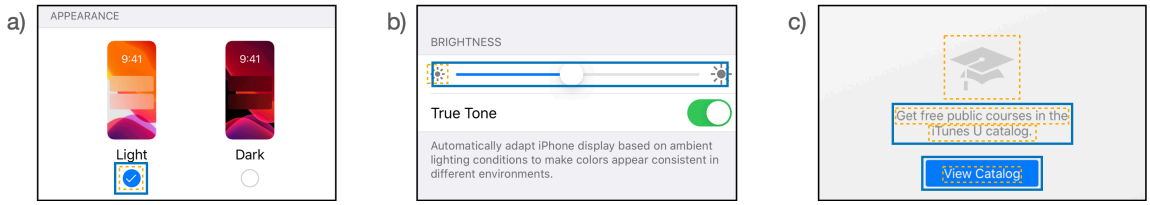

Fig. 1. a) Containment: having at least 85% of annotation box (in orange) area lies within UI element's bounding box (in blue). b) Overlap: the intersection over union (IoU) of annotation box (in orange) and UI element's bounding box (in blue) is at least 5%. c) The top annotation box (in orange) is not contained by or overlapped with any UI element's bounding box (in blue).

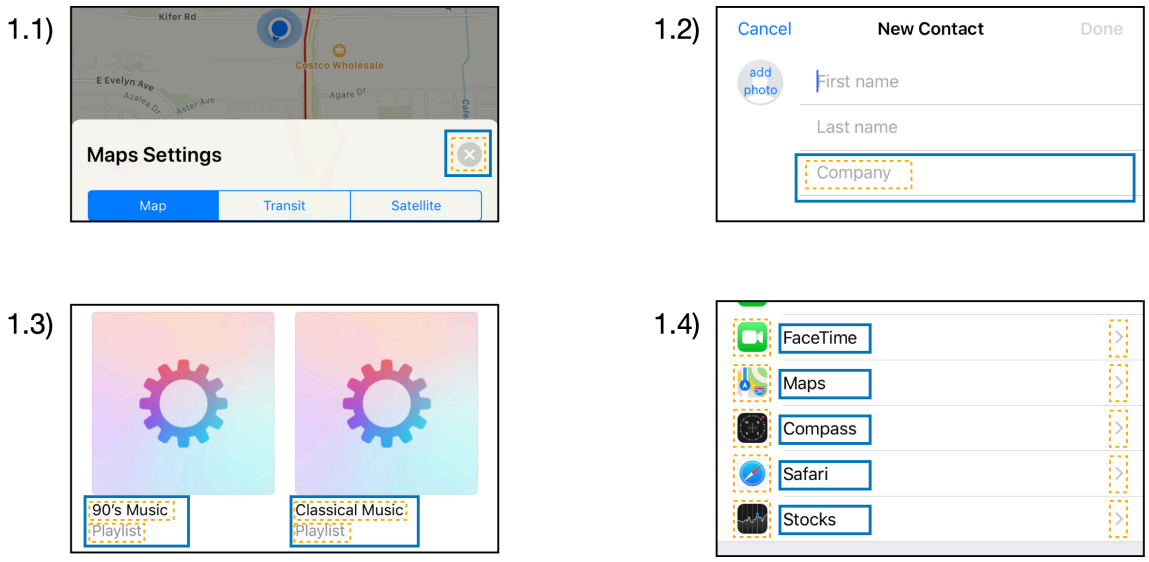

Fig. 2. (1.1) For "Close" Icon, the UI element's bounding box (in blue) is looser than its annotation box (in orange). (1.2) For "Company" Text Field, the nested outline and placeholder annotations (in orange) should match with UI element (in blue). (1.3) For two-line text, the UI element's bounding box (in blue) encapsulates two smaller annotation (in orange). (1.4) For each table cell, the unmatched Icons (in orange) horizontally align with matched Text annotations (in blue).

1.2 Nested Annotations

Some types of nested annotations (annotations that contain other annotations) can correspond to just one UI element, as documented below. If any annotation within one of these nestings matches with a UI element, then all annotations within the nesting match with that UI element.

Some of these nested annotations have the same UI type, but differ in their additional information mentioned in Paper Section 3.2. A Text Field can include an annotation for its placeholder or entered text in addition to an annotation for the whole element. Similarly, Sliders can have separate annotations for its handle and the whole element. A Segmented Control or Tab Bar Item can be composed of individual annotations for its border, text, and/or non-textual components.

Many, though not all, the Container annotations can represent one UI element. The most common of these cases are Containers that represent buttons or table cells, which for our heuristics we define them to be composed of at least one Text annotation and at most two Icons and/or Pictures. Other cases include a Container annotation with one Checkbox and one Text annotation, or a (potentially nested) Text Field with at most two additional Text annotations (often times labeling the Text Field).

We included two other heuristics for Container annotations. If all of a Container's nested annotations have matches, then the Container annotation also has a match. Similarly, if none of its nested annotations have a match, then the Container does not have a match. Because these heuristics depend on whether or not nested annotations have matches, we apply them both after the heuristics in Appendix Section 1.1 and again after applying all heuristics.

1.3 UI Elements Containing Multiple Annotations

Some UI elements contain multiple annotations. For example, a UI element for a large piece of text could encapsulate multiple smaller Text annotations for its individual lines or paragraphs. For this case, we match each of these contained annotations with this UI element if all the contained annotations are non-clickable Text. Some UI elements contain what would otherwise be one of the nested annotation cases described in Appendix Section 1.2, with deviations like a missing annotation for a Container or border. If a UI element's contained annotations fall into this category, they also match to this UI element.

Before applying these heuristics, we applied more generic ones as follows. If all its contents have matches, we ignore this UI element; it won't increase the number of matches. If all but one of its contents have matches, we match the remaining annotation with this UI element. Otherwise, it's worth checking whether this UI element's bounding box contains other UI element bounding boxes. If so, make sure to recursively examine these contained UI elements first.

1.4 Ignore-able Annotations

From an accessibility standpoint, leaving out some specific types of UI elements, while not always ideal, would not effect the functionality of the app. To avoid counting such cases of unmatched annotations as missing UI elements, we developed heuristics to discount or ignore them.

We ignore unmatched Icons that horizontally align with matched Text annotations that are not independently clickable, as is the case for decorative Icons in table cells or buttons without a Container annotation. Similarly, we discount potentially decorative Pictures by ignoring Picture annotations that either that have a small bounding box (area < 3600 pixels) and are not clickable, or contain one or more nested annotations (as it may be a background image).

We also ignore buttons added to the status bar that are not part of the app. Specifically, we ignore annotations for the "return to previous app" button that iOS adds to the left part of the status bar when you launch an app from within a different app (launching a downloaded app from the App Store). The heuristic here is to ignore any unmatched annotation that is an Icon or Text and has a bounding box within empirically derived thresholds representing that location and approximate size.

2 DETAILS OF WORKERS IN DATA COLLECTION AND ANNOTATION

2.1 Data Collection

We hired 10 workers in the U.S. to collect the iOS app screen dataset. All of them are English speakers and smartphone users. They all have done crowd-sourcing tasks before, but did not have experience with app screen collection tasks. It took a total of 1,692 person-hours to collect this dataset, not including onboarding time (three of the authors spent a few hours to onboard all workers).

2.2 Data Annotation

We hired 40 workers outside the U.S. to annotate the iOS app screen dataset. All of them are English speakers and smartphone users. They all have done crowd-sourcing tasks before, and had experience with our internal annotation toolkits. It took a total of 5,700 person-hours to annotate this dataset, not including training time (a crowd manager spent about half a day to train all annotators). The instructions to train annotators are attached as a separate Supplemental Material (removed instructions that contain confidential information, e.g., how to use internal annotation tool).

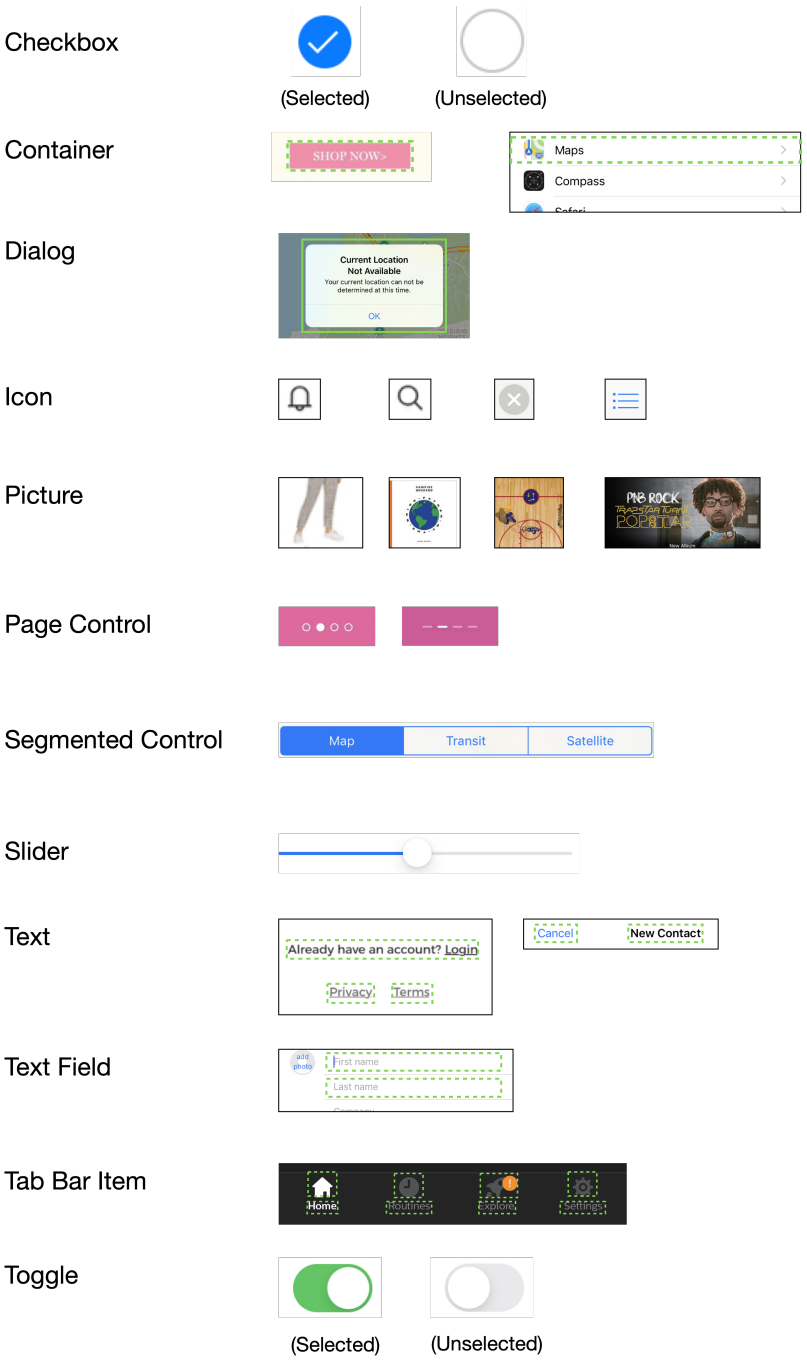

Fig. 3. When annotating UI elements, for each bounding box, annotators assigned one of 12 common UI types based on visual inspection. Here are examples of these UI Types.