Smart Display Accessibility Mode (S-DAM)

An Programmatic Approach to Improve Accessibility on Digital Displays for Light and Contrast Sensitivity

ABSTRACT

Contrast or Light Sensitivity is a common condition that affects most people and increases with age. iOS & Android mobile platforms have display accessibility features like high contrast, invert colors, smart invert and color blindness filters to provide ease of access for users with various vision impairments. These display accommodations either modify the entire screen basen on some simple rules or rely on app developers to use specialized APIs. For example "Smart Invert" provided by iOS requires app developers to explicitly tag content to be inverted or skipped. This paper provides an algorithm that is capable of analyzing, identifying, and segmenting screen content to transform an screen's image into a more accessible form. By ignoring media like images and video, this technique provides an option for users with contrast or light sensitivity to view graphical content in their original form, thus providing them with an experience similar to what the app developers intended for a user without any vision impairments. We provide a platform agnostic solution called Smart Display Accessibility Mode (S-DAM) that does not require extensive app developer input and compare our solution to existing display accommodation features provided on iOS and Android mobile platforms. We also provide a case study on the limitations of some of the currently available display accommodation features in iOS and Android mobile platforms against several popular mobile apps from a variety of app categories.

KEYWORDS

Digital Accessibility, Vision Impairment, Smart Invert, Dark Mode, Light Sensitivity, Contrast Sensitivity

1 Introduction

Electronic displays are ubiquitous in our day-to-day lives. It is safe to say that most lifestyles require access to a smartphone, or require frequent interaction with a screen. From wristwatches to cars, digital displays have replaced analog displays, and this trend does not show signs of slowing down.

The luminance or brightness in an electronic display can hinder the reading ability or cause pain to people with visual impairments due to light or contrast sensitivity. Individuals with light sensitivity may experience eye pain or headaches when viewing a bright screen. A bright screen can be a result of a high system level brightness setting compared to the ambient lighting conditions, an app with predominantly white pixels, or both. These conditions require an impacted user to change the display brightness or switch to dark mode. Contrast sensitivity is the ability to differentiate light and shaded areas of a screen. For example, dark grey colored text on a white background can be challenging to read for someone with low contrast sensitivity. The white background overpowers, or blows out the darker text it contains making it strenuous and difficult to read. These visual impairments are due to health conditions, congenital disabilities, and have been found to decline with aging [2]. In a fact sheet, the World Health Organization (WHO) [1] mentions "age-related macular degeneration" as the third leading cause for vision impairment globally. As the first major smartphone-using generations age, it is increasingly important for electronic displays to provide accommodations for users with light or contrast sensitivity.

Major mobile platforms iOS and Android provide a variety of accessible display options for light or contrast sensitive users like dark mode, night mode, invert, smart invert (iOS), high contrast, etc. Most of these accessibility features work effectively on user interfaces (UI) developed by the platform creators (Apple and Google respectively) but are ineffective on UI developed by third party app developers. The accessibility solutions provided by platforms either apply an unsophisticated filter to the displayed pixels or depend on app developers tagging the contents correctly and designing UI with the correct contrast ratio. Many apps use WebView based frameworks that embed rich HTML content in an app since web content is adaptive to all platforms and offers the convenience of a common codebase for all platforms. But WebViews typically lack native integration with system level accessibility features required for light or contrast sensitivity. There is a need for a standard solution that does not rely on app developers understanding and solving accessibility issues themselves and is agnostic to complex mobile platform components like Web View.

2 Existing Solutions

The existing solutions for users with light or contrast sensitivity cover several needs, but no single solution or combination of solutions offers a complete solution. We will now cover the capabilities of each existing solution and their shortcomings. Table 1 provides a list of popular mobile applications and their support for iOS dark mode and smart invert.

2.1 iOS / Android Invert

A basic invert mode is the simplest of all the solutions and has been available for the longest time. Invert takes the entire screen and flips each pixel to the opposite color. For example, in terms of RGB, pure white with values (255, 255, 255) will be inverted to pure black with values (0, 0, 0). Media content like images and video are significantly distorted often to the point of being unrecognizable. While a light themed app's text content when inverted is now ideal for individuals with light or contrast sensitivity, inverted media content really takes away from the user experience. Also invert mode is not necessary on apps that support a proper dark mode. Because of this, users often use a shortcut to toggle invert mode which is yet another unnecessary accessibility barrier. More recent existing solutions improve upon this implementation.

2.2 iOS Smart Invert:

Smart invert, the next major advancement relevant to light and contrast sensitivity, arrived in fall of 2017 with Apple's iOS 11. Smart invert aims to solve the issue of distorted media content by allowing app developers to mark UI elements to be ignored for inversion [8]. Technically this solution can be near perfect, allowing users to view their general UI content in the way that is most comfortable to them without any of the media distortions introduced by the basic invert mode. There is no need for the OS or some algorithm to be able to infer what should or should not be inverted because the app developer has intimate knowledge of their UI components and can precisely specify rules for inversion. However, we found that most app developers do not support this feature despite smart invert being available for over 4 years. See attached images at the end of the paper for several examples of smart invert. In the Evaluation section we will dive deeper into the spotty implementation and bugs with smart invert. Popular web browsers and WebView based apps also don't take advantage of smart invert when rendering img tags. Similar to the basic invert mode, smart invert can also be assigned to a shortcut on iOS. It is worth noting that some apps have a buggy implementation where the invert only applies correctly if the app is launched in the accessibility mode. If the mode is toggled with the app open invert is improperly applied. Despite the lack of support and buggy implementations from app developers, smart invert is still strictly a better solution than the basic invert mode.

App Category	App Name	Dark Mode	Smart Invert
Life Style	Yelp	Not Supported	Inverts Media
	Uber Eats	Not Supported	Inverts Media
Social	Reddit	Supported	Inverts Media
	Instagram	Supported	Supported
	Twitter	Supported	Some Media contents are inverted
Financial	Paypal	Not Supported	Some Media contents are inverted
WeView	Chrome Browser	Not Supported	Inverts Media

Table 1: **Support for Dark Mode and Smart Invert in popular mobile applications.**

2.3 iOS / Android Dark Mode:

The availability of dark mode was a major step forward for light and contrast sensitivity. Dark themes have been commonplace for app developers in their tooling and IDEs for a very long time. App developers implemented dark themes on their apps well before system level support was introduced by Apple or Google. For example, popular apps such as Reddit and Twitch.tv have had a dark theme since as early as 2017. At the system level, in 2010 Windows Phone was one of the first mobile platforms to introduce a dark mode. iOS and Android followed much later with a system level dark mode in fall of 2019. A well implemented dark mode is the best existing solution for users with Light or Contrast Sensitivity. In dark mode, the entire system and all apps that support it switch to a predominantly light text on a dark background color scheme in contrast to the traditional dark text on a light background. Similar to smart invert, dark mode faces the same issues with app developer support. Tapping into the system level dark mode requires an app developer to implement all the switching logic to recolor their UI to a different theme in response to the system settings. Many apps either do not implement any color theming at all, or haven't been written to take advantage of the system level APIs even if they have implemented a dark theme themselves. Because of this, even with the advent of dark mode, users with light or contrast sensitivity still have to rely on the older color inversion based techniques to use their devices.

2.4 iOS / Android Night Mode:

An honorable mention for Night Modes. Known as 'Night Shift' on iOS or 'Night Light' on Android, this mode will increase the warmth or temperature of the display. This reduces the amount of unnatural 'blue' light emitted by a display by giving the entire screen a yellow-orange tint - meant to emulate the warm light emitted by fire or traditional light bulbs. This feature does not directly resolve the problem for Light or Contrast Sensitivity, but it does slightly lessen the impact of a bright and mostly white display and reduces eye strain.

3 Smart Display Accessibility Mode (S-DAM)

Recent advancements in smartphone chipsets have cleared ways for advanced on-device processing capabilities; even a mid-range smartphone today can run machine learning algorithms fast enough for real world use. Our solution, Smart Display Accessibility Mode (S-DAM), provides a programmatic technique that is platform agnostic and is purely based on content in the current active view and can tap into the aforementioned available processing power.

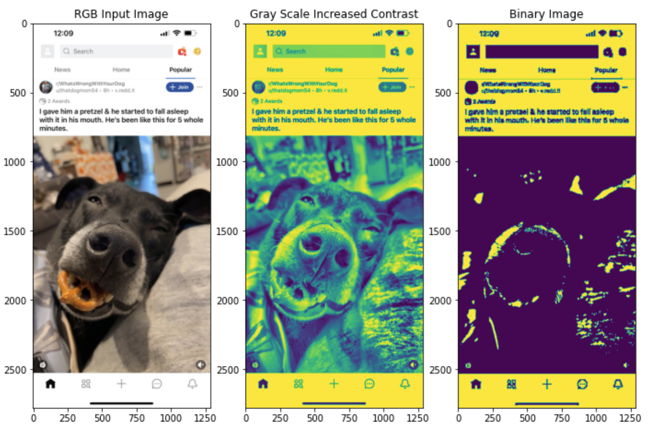
Our algorithm analyzes a screenshot and converts the input image to a light and contrast sensitivity accessible image by emulating a dark mode, improving the contrast all while detecting and avoiding content like media that is best viewed in its original form. The algorithm consists of three main steps; the first step is called "Detect Media Content," scans the currently active view to identify sections to modify contrast ratio or reduce brightness and generate a bitmask of media content to ignore. The second step, called "Emulate Dark Mode", will take light themed screens and generate a convincing dark mode equivalent. The third step, called "Improve Contrast," will take the dark themed screen and make dark pixels darker while maintaining clarity in text and media. We used python for S-DAM, a link to the Github repository is provided in reference section [10].

3.1 Step 1 (Detect Media Content)

The initial step of our solution is to analyze the screen and identify media content. For analyzing and contouring the screen content, we decided to use the OpenCV library [4]. OpenCV is a well-known real-time vision library implemented in C++ and has library interfaces for C++, Java, Python, and MATLAB; we used the Python libraries for our implementation. The active frame of the current screen is taken as an image in GrayScale format and converted to a two-dimensional array using the "cvtColor" function. The image is converted to grayscale so that it is easier to manipulate and process content.

3.1.1 Image Processing:

The primary goal of image processing is to enhance the media content in the input image, which helps bring out the difference between media content and the material UI generally used in a mobile application. The logical first step of processing an image is to improve contrast without increasing the noise or losing information in an image to detect media content. For this, we use the "createCLAHE" function with defaults from OpenCV that applies an adaptive histogram equalization [5]. The increased contrast will bring out more detailed information of the structure in the media content. High contrast could push the media content to the darker spectrum, Gamma correction helps enhance any dark media content. We used the "LUT" function of OpenCV that provides gamma correction based on a color map lookup table. A colormap is mapping for 256 color values to numbers in the range 0-255; the media content in the input image are pushed to a lighter spectrum using a gamma value greater than one in the colormap lookup table.



I gave him a pretzel & he started to fall asleep with it in his mouth. He's been like this for 5 whole minutes.

Figure 1: **Image processing from RGB to Grayscale image with improved contrast and gamma correction and finally the binary output.**

At this stage, with the media content at its best clarity, we now segment the image by using a simple threshold function from OpenCV, with a threshold value of 127. This function scans through the image pixel by pixel and compares the pixel intensity to the threshold value. If the value is less than the threshold, it converts the pixel to black; otherwise, it converts it to white. The final output of the image thresholding is a binary image with just black and white pixels. The binary image could have noise from content like text or even distorted media content. Two OpenCV functions, "erode" and "dilate," are used to apply a morphological transformation [6] to reduce the noise in the binary image. A sample output of the image processing steps is shown in Figure 1. The grayscale increased contrast output shows the increased detail of the media content especially on the beige colored blanket behind the dog, without this processing the blanket could have been considered as a part of the white UI background.

3.1.2 Contour and Create Bitmask:
Now for the final processing of Step1, we find the regions of media content and create a bitmask, a sample output of which is shown in Figure 2. We utilized the "findCountours" function from OpenCV to identify the contours of media content; the input for this function is a binary image. In the processed binary image, most media regions will have a black pixel value. We noticed that the OpenCV is able to contour white pixels much better than black, so we inverted the pixels of the binary image before performing the contouring. The media content pixels were inverted to have a white pixel value, and the background, which in this case, mobile application UI components, will have a black pixel value. We chose to draw the contours in a rectangle shape because most media contents are displayed in a rectangular form . Also, we choose to ignore contours of area less than 90000; the area value is one of the program parameters that can be adjusted based on the screen type. Using the contours identified a bitmask of the size of the input image with false for pixels with media content.

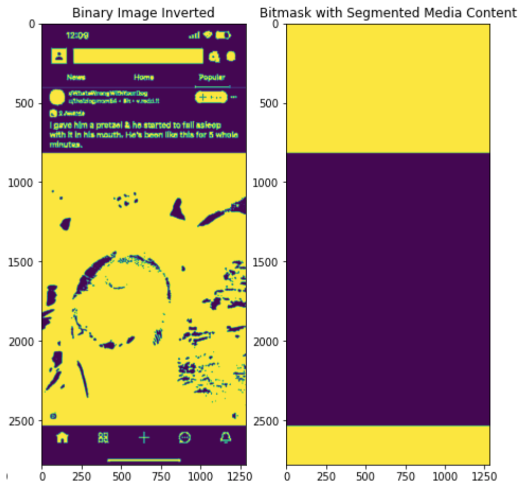


Figure 2: **Inverted binary post image processing to bitmask generated using contours detected by OpenCV.**

3.2 Step 2 (Emulate Dark Mode)

In this stage we take the screen content and transform it to emulate a dark mode. If the input image is already in a dark theme then we skip this step. There are two inputs to this step, the unprocessed image and the bitmask generated in Step 1. We will take advantage of the HSL color formatting scheme to simplify this process. We are specifically interested in the S (saturation) and L (lightness) numbers. Saturation describes how rich or vivid a color is. Lightness describes how much light or intensity to draw the color with. S-DAM uses saturation to determine whether or not a pixel is colored. When saturation is low this guarantees the pixel is grayscale or close to grayscale independent of the H or L values. If S is not close to zero, the pixel may still be close to grayscale if the L value is close to 0 (min indicating black) or 255 (max indicating white).

We make some assumptions in our implementation that app developers will follow best practices for color choices and not design an app that is difficult to use even for individuals without impaired vision. This means on light themed apps we expect light backgrounds close to white and text that is relatively close to black. If the app uses shades of gray over other shades of gray, then S-DAM will perform unpredictably.

To determine if a screen's non-media content is in light or dark mode we check the average L value of all the unmasked pixels. If the average L is less than 128 (half of the range) then we proceed with dark mode emulation. Otherwise we skip to Step 3.

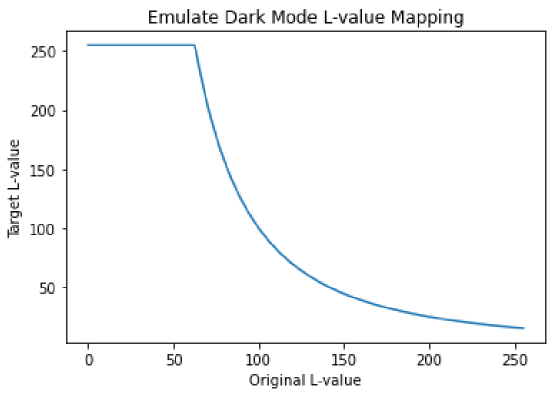
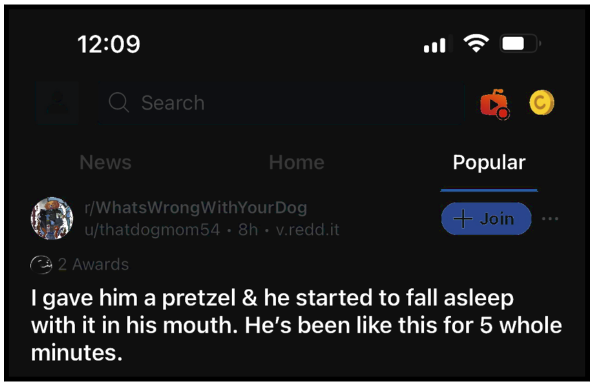


Figure 3: **Mapping function used to modify L-values on non-media pixels in an HSL input image to generate an Emulated Dark Mode image.**

To transform the image to dark mode we take the entire screen pixel by pixel and apply a function that maps the HSL value for each light mode pixel to its dark mode equivalent. On top of the bitmask, we also consider whether a pixel is colored or close to grayscale. If a pixel is close to grayscale we apply the L value inverting function (1) visualized in Figure 3.

$$target_L = min\left(255,\ \frac{1{,}000{,}000}{original_L^{\ 2}}\right) \qquad (1)$$

This function (1) is designed as a continuous mapping that favors making pixels darker and results in an image as seen in Figure 4. Only a modest band of originally dark pixels (those with small L values) will be mapped to white while preserving greys in the middle just in case the app developer needed that color. Because valid values for L are bound between 0 and 255, we cap the max output value to 255.


12:09

Search

News Home Popular

r/WhatsWrongWithYourDog
u/thatdogmom54 • 8h • v.redd.it

Join

2 Awards

I gave him a pretzel & he started to fall asleep with it in his mouth. He's been like this for 5 whole minutes.

Figure 4: **Cropped result of Step 2 (Emulate Dark Mode) on the header portion of a sample Reddit app screen.**

3.3 Step 3 (Improve Contrast)

Finally we take the dark themed screen and push the contrast all the way to the levels of a true black background theme. The goal is to fix all dark background pixels to a true black RGB value of (0,0,0). This serves to further reduce overall emitted light and takes S-DAM a step beyond most dark theme implementations that employ many shades of grey. On modern OLED displays a pixel with RGB color (0,0,0) emits no light, all diodes are turned off, but as soon as any single channel of RGB requires a positive value, the pixel must light up to show color even if in shades of grey. Taking advantage of turning pixels off can have a very significant impact on overall light emitted.

$$target_L = 208 \,/\left(1 + e^{\left(40 - original_L\right)/4}\right) \qquad (2)$$

Like the previous step there are two inputs, the HSL formatted output from Step 2, and the same bitmask. A sigmoid function (2), visualized in Figure 5, is used to map unmasked pixels to their final colors. The target L-value is bound between 0 and 208 instead of the max value of 255 to keep the total light emitted down without making white pixels so dim that they cannot be easily read. The value 40 in the function is used to place the cutoff where pixels with L less than 40 will tend towards black, and the rest will tend

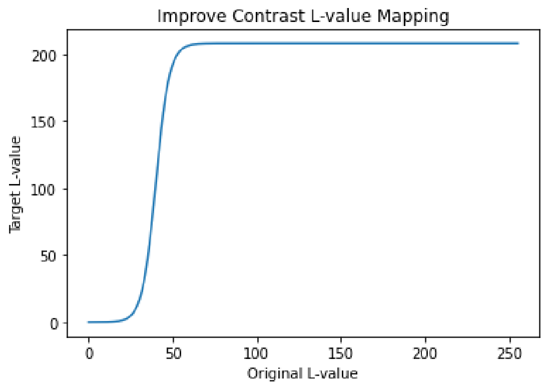
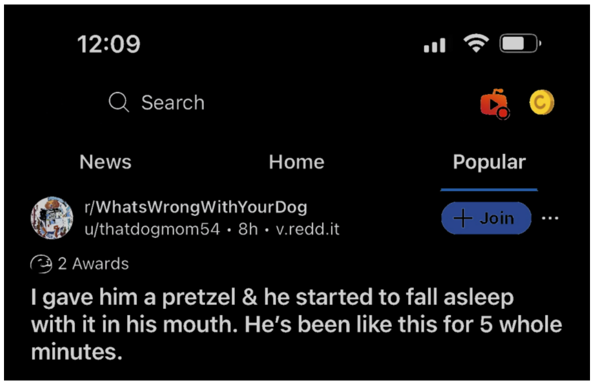


Figure 5: **Mapping function used to modify L-values on non-media pixels in an HSL input image to improve the contrast of a dark themed screen.**

towards white. The value 4 dictates how sharply the cutoff transitions. 4 was chosen because it is a relatively sharp cutoff, but still allows S-DAM to preserve some grey tones. The net result of this function is that dark pixels will become darker and light pixels will become lighter. See Figure 6 with the output of the Improve Contrast step in contrast to Figure 4 from Step 2. Notice the difficult to read


12:09

Search

News Home Popular

r/WhatsWrongWithYourDog
u/thatdogmom54 • 8h • v.redd.it

Join

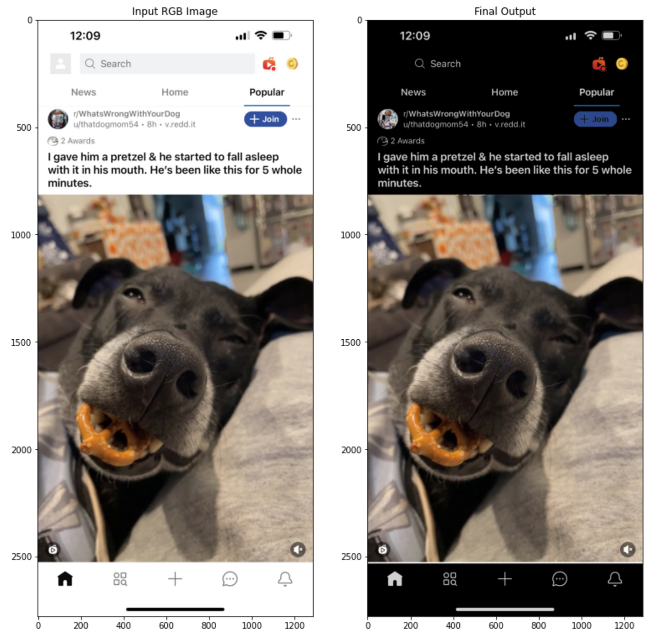
2 Awards

I gave him a pretzel & he started to fall asleep with it in his mouth. He's been like this for 5 whole minutes.

Figure 6: **Cropped result of Step 3 (Improve Contrast) on the header portion of a sample Reddit app screen.**

gray pixels are now a stronger white and that any dark grey artifacts visible around the search input are now pure black. Details like the battery outline which is difficult to see in the output of Step 2 are also now visible again.

In practice Steps 2 and 3 are executed in one iteration of all the screen pixels instead of two separate loops as a small performance optimization. Figure 7 showcases the end result of S-DAM against the original screen image. In this particular example all text is successfully converted. The main image content is successfully preserved. Icons have retained their coloring. There are some artifacts introduced at the boundaries of the main image and around the colored icons. The smaller icon-size image at the top left was inverted partially, and the '+ Join' button would still be more legible with white text. We will analyze the accuracy of S-DAM in more depth in the Evaluation section.



I gave him a pretzel & he started to fall asleep with it in his mouth. He's been like this for 5 whole minutes.

Figure 7: **A side by side view of the input and the final output of S-DAM.**

4 Evaluation

To evaluate the performance of S-DAM we have selected a handful of popular apps distributed across several major app categories; like lifestyle, social, financial. To gauge the impact for individuals with contrast or light sensitivity, we place an emphasis on the effective light output, accuracy, and the contrast ratio between background and foreground.

App Category	App Name	Original	Dark Mode	Smart Invert	S-DAM
Life Style	Yelp	213.58	N/A	41.02	81.65
	Uber Eats	201.71	N/A	73.01	135.16
Social	Reddit	164.70	75.53	90.79	74.81
	Twitter	201.56	56.93	60.14	91.19
Financial	Paypal	219.66	N/A	35.33	114.30
WeView	Chrome Browser	184.93	152.04	69.49	61.11

Table 2: **Effective Light Output scores (lower is better) for various popular apps in different existing accessibility modes compared to S-DAM.**

4.1 Effective Light Output

The effective light output is a metric designed for this paper that aims to quantify the total amount of light emitted by all the pixels on a display. To calculate effective light output we will again rely on the HSL color scheme and take an average of the L value across all pixels. To ensure fair comparisons all screenshots have been taken from the same device (iPhone 13 Pro Max) for a constant image size and consistent OS. Light sensitive users prefer a display that emits less overall light, so a lower effective light output score is better.

Different display technologies scale differently in terms of their practical benefit at a given effective light output. For example, OLED displays benefit far more from a pixel with an L value of 0 than an LCD display. However, because we are approaching the problem from a platform and technology agnostic standpoint, and because even on LCD displays a lower effective light output score is better, we will not be considering display technology in this evaluation.

The effective light output scores applications are shown in Table 2 for each tested app. The results reveal some interesting trends and a couple weaknesses of S-DAM. The apps that support a dark mode all performed very well with the exception of Chrome. Chrome, which is based on WebView technology, did not take any measures to invert the web page contents. It is clear that with proper support for a dark mode, the bulk of accessibility concerns for light

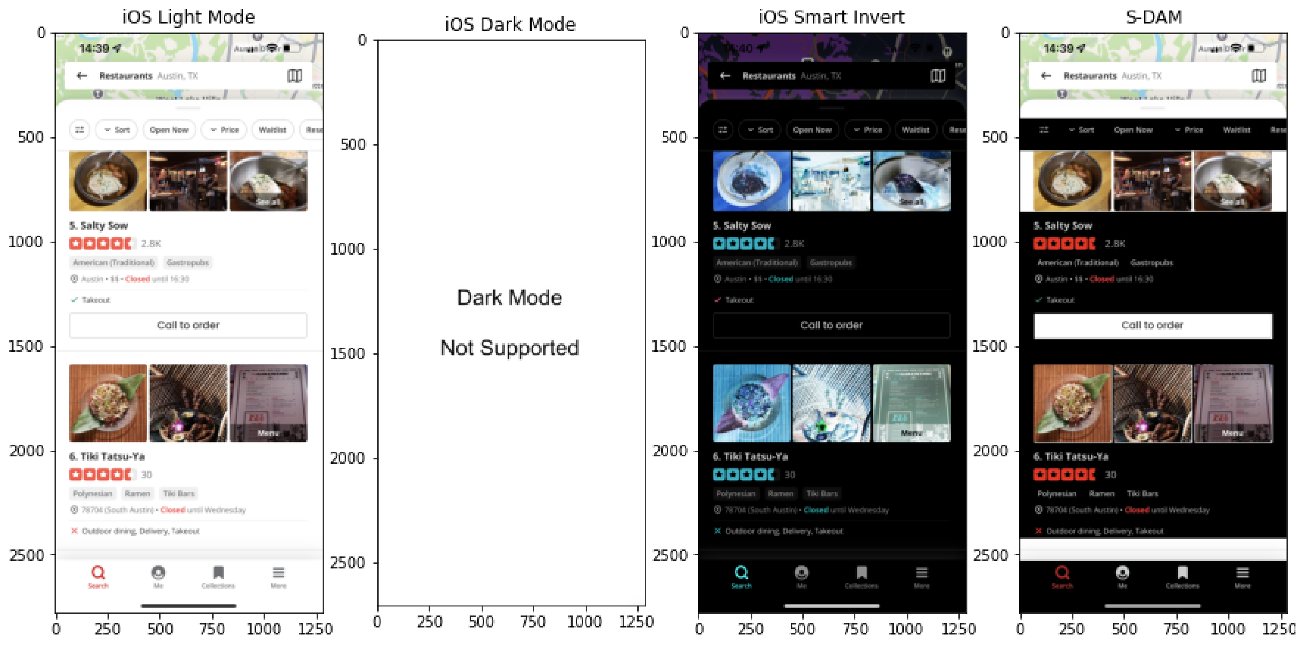


Figure 8: **A side by side view of screenshots from the Yelp app in Light Mode, Smart Invert, and S-DAM.**

and contrast sensitivity is solved. S-DAM on top of the dark mode screens as an input (not shown in Table 2) would achieve only marginal gains by capping the lightness of the white text to 208 out of 255; this may be valuable to light sensitive users. We can also infer from the results that S-DAM's performance relies heavily on how well it contours and segments the screen contents. For example, in Figure 8 we see that several elements of the Yelp app were not segmented properly resulting in blocks of white and/or light pixels. However on apps where S-DAM segmented very well, like in the Reddit app, the effective light output can be better than Dark Mode.

4.2 Accuracy

We evaluate accuracy based on the frequency of errors in the post-processed screen. An error is either a section of media that did not retain its original coloring or a region of UI that was not correctly converted to a dark theme. Nearly all errors originate from incorrect decisions in the contouring phase. OpenCV has limitations when handling contents that are different shades of the same color. S-DAM output exhibits several limitations in Figure 8. At the bottom there is a drop shadow just above the navigation buttons along the bottom of the screen. The OpenCV contouring library aggressively cuts off at the gradient and mislabels it as media content. The 'Call to order' button was successfully segmented, however because the overall size of that segment was less than our threshold of 90000 pixels, it was not considered as part of the invertible UI. Also, because we used rectangle-shaped contours, white pixels are displayed at the corners of every image on the Yelp screen or the corner is failing to be detected as an edge. A more exaggerated example of rounded corners interfering with S-DAM is visible at the top of the Yelp screen where the topmost contoured section cuts off at the very bottom of the curved element. The coloring of the Yelp stars and text content are emulating a dark mode very well and legibility is great. Compared to the Smart Invert result, no images are distorted which contextually is very important in the Yelp app.

Several more examples are attached at the end of this paper for evaluation. Some notable observations. Smart invert occasionally only partially works as seen in the Uber Eats (Figure 10) and Twitter (Figure 11) apps. S-DAM outperforms smart invert when it comes to preserving the color of buttons and icons as seen in the Twitter (Figure 11) and Reddit (Figure 12) apps.

4.3 Contrast

Contrast is difficult to evaluate, but Google provides guidelines [7] on how to measure whether or not the contrast between background color and text is suitable for digital screens. Contrast is scored on a range from 1 to 21 where 1 is failing. A score of 21 is achieved with pure white on pure black or the opposite. Because S-DAM performs a cap on white pixels (at 208 out of 255) in Step 3 of the implementation, by Google's metrics our approach will max out at a score of 13.61 which still passes all required criteria. In the context of light sensitivity a 'perfect' score of 21 actually is not ideal. At such high contrast levels a light sensitive user will see the white light bleed over and into the other colored or black pixels. Think of it like trying to read the specs printed on a lightbulb while the bulb is illuminated. The light from the bulb would completely wash out the specs text. S-DAM performs the role of shaded lenses, enabling light sensitive users to consume regularly retina roasting content. However the contrast cannot be reduced too much or text will become illegible against its background and will become difficult especially for contrast sensitive users. Therefore we split the difference and landed on a moderate contrast in the middle of the acceptable range.

5 Future Work

S-DAM is a fantastic step in the right direction but it leaves a lot of room for improvement and it opens up many other interesting topics to explore. The most challenging component and the most obvious weakness of S-DAM is the contouring logic that detects media content to be ignored by the Emulate Dark Mode and Improve Contrast steps. There may be room to train an ML model specifically optimized to locate images within an image. The implemented process could also be modified with a secondary contouring phase to remove noise from the first pass. A second contouring phase could also be used to detect smaller images or icons that are currently ignored by the minimum region size requirements. There is also a need to prototype S-DAM on an actual device to evaluate the real-time performance, the energy consumption, energy consumption compared to Smart Invert, and other operating metrics.

We also believe that a Smart Invert type of feature for HTML browsers may be low hanging fruit for the major platform owners, Apple or Google, or even third-party developers to improve upon via browser extensions. A perfect solution may not be worth pursuing, but at the very least elements in the HTML DOM with an img tag could be excluded from inversion while inverting everything else.

Another future work is possibly to tune the constant values we have used in the algorithm, like the minimum area of the contours to consider for images and the threshold limit for binary image segmentation. These constant values can be further refined to be adaptive based on the screen type that is processed. For example, a social feed screen will contain large images compared to a catalog screen which will contain images of constant sizes.

Finally, S-DAM is completely platform agnostic. While this paper has been in the context of mobile applications, a natural first step should be to verify the technique on a PC. All the same challenges and accessibility needs are present without the limitations of mobile operating systems, software restrictions (like on iOS), with much more powerful hardware, and so on.

6 Conclusion

S-DAM performed well by reducing overall emitted light and tuning the contrast of various mobile applications to suit the needs of individuals with light or contrast sensitivity. We believe these visual impairments will become increasingly problematic as entire generations that are accustomed to using digital displays are growing older. S-DAM was able to handle most screens with passable issues primarily related to the algorithms we implemented and used to detect media content. We believe that the image segmentation algorithm has the most room to improve and can be tuned to achieve a better output. S-DAM has the potential to be a significant enhancement for users who have light or contrast sensitivity and already outperforms some apps. It is only a matter of time and effort to turn S-DAM into something very powerful.

REFERENCES

[1] WHO Blindness and vision impairment fact sheet : https://www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment
[2] Contrast sensitivity decline with ageing: a neural or optical phenomenon : https://pubmed.ncbi.nlm.nih.gov/3454919/
[3] Usability.gov is the leading resource for user experience (UX) best practices and guidelines: https://www.usability.gov/about-us/index.html
[4] OpenCV is a library of programming functions mainly aimed at real-time computer vision : https://opencv.org/
[5] OpenCV Contrast Limited Adaptive Histogram Equalization : https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html
[6] OpenCV Morphological Transformations: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html
[7] Google. Android Accessibility Help: https://support.google.com/accessibility/android/answer/7158390?hl=en
[8] Apple. Developer Documentation: https://developer.apple.com/documentation/uikit/uiview/2865843-accessibilityignoresinvertcolors
[9] OpenCV Contour Detection using python library and its limitations : https://learnopencv.com/contour-detection-using-opencv-python-c
[10] Github Repository for S-DAM code and screens used for evaluation - https://github.com/friedliver/smart-accessibility-modes-for-mobile-display

RESULTS

Paypal

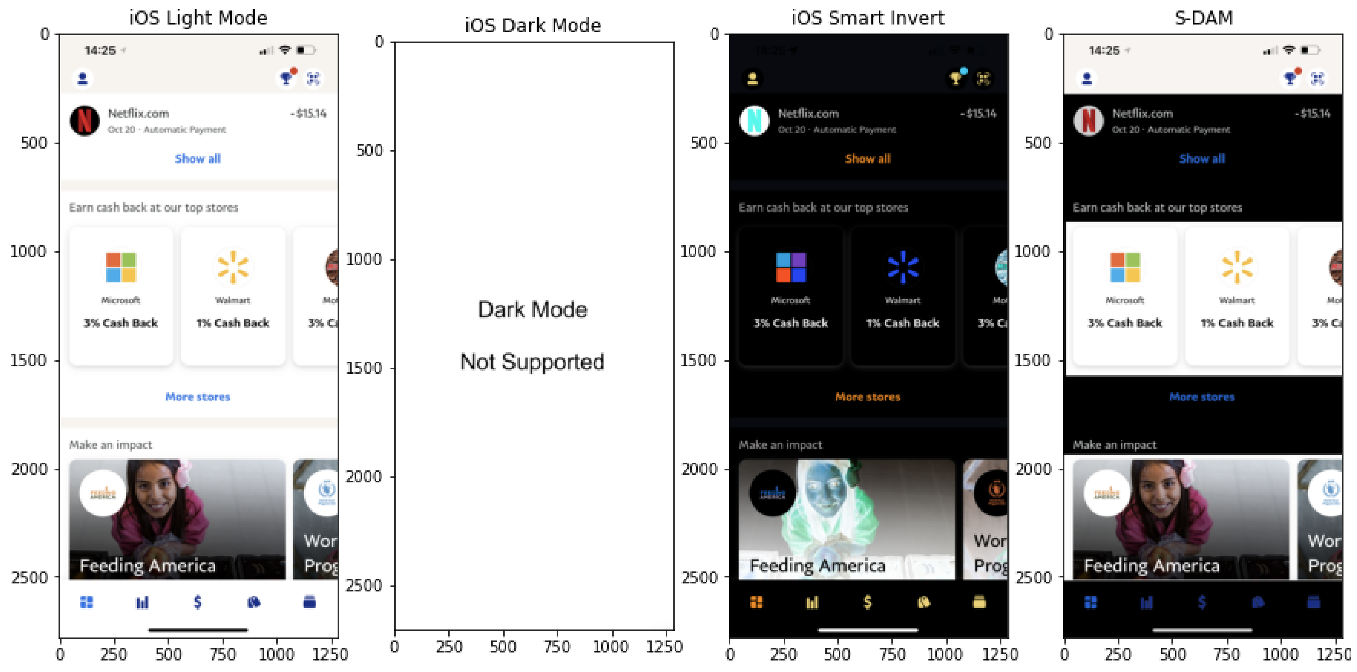


Figure 9: **A side by side view of screenshots from Paypal app in Light Mode, Smart Invert, and S-DAM.**

UberEats

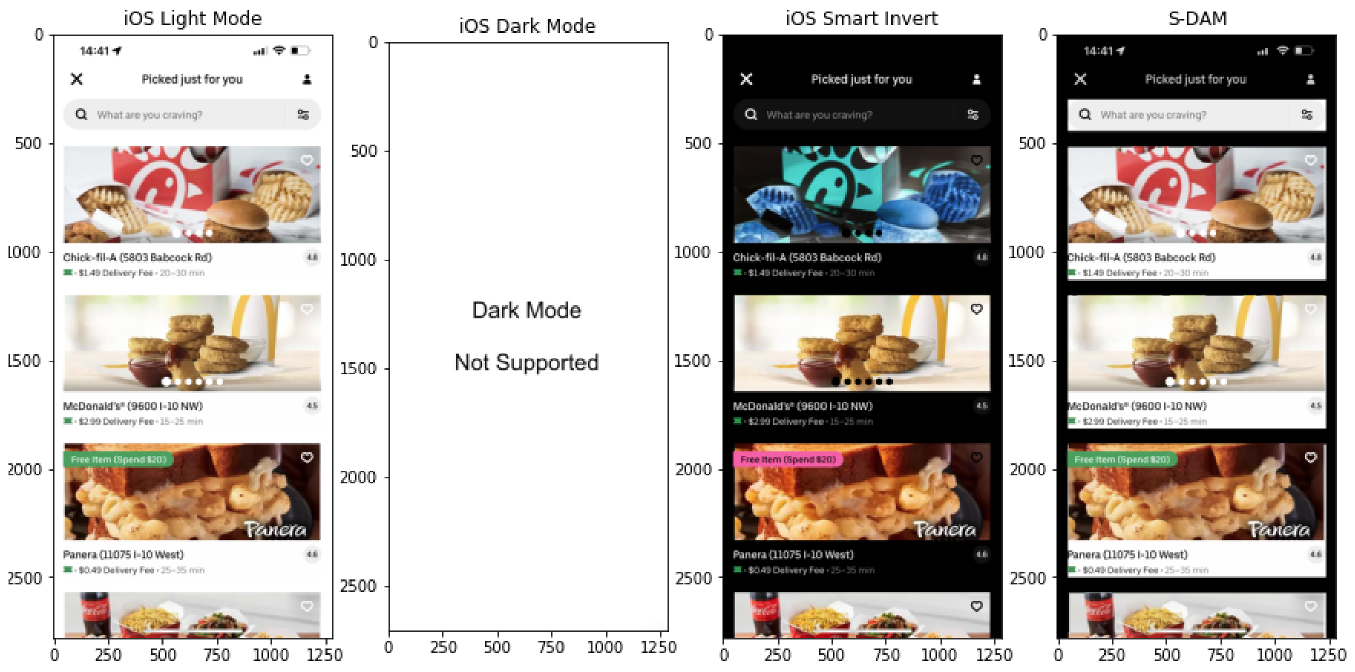


Figure 10: **A side by side view of screenshots from Uber Eats app in Light Mode, Smart Invert, and S-DAM.**

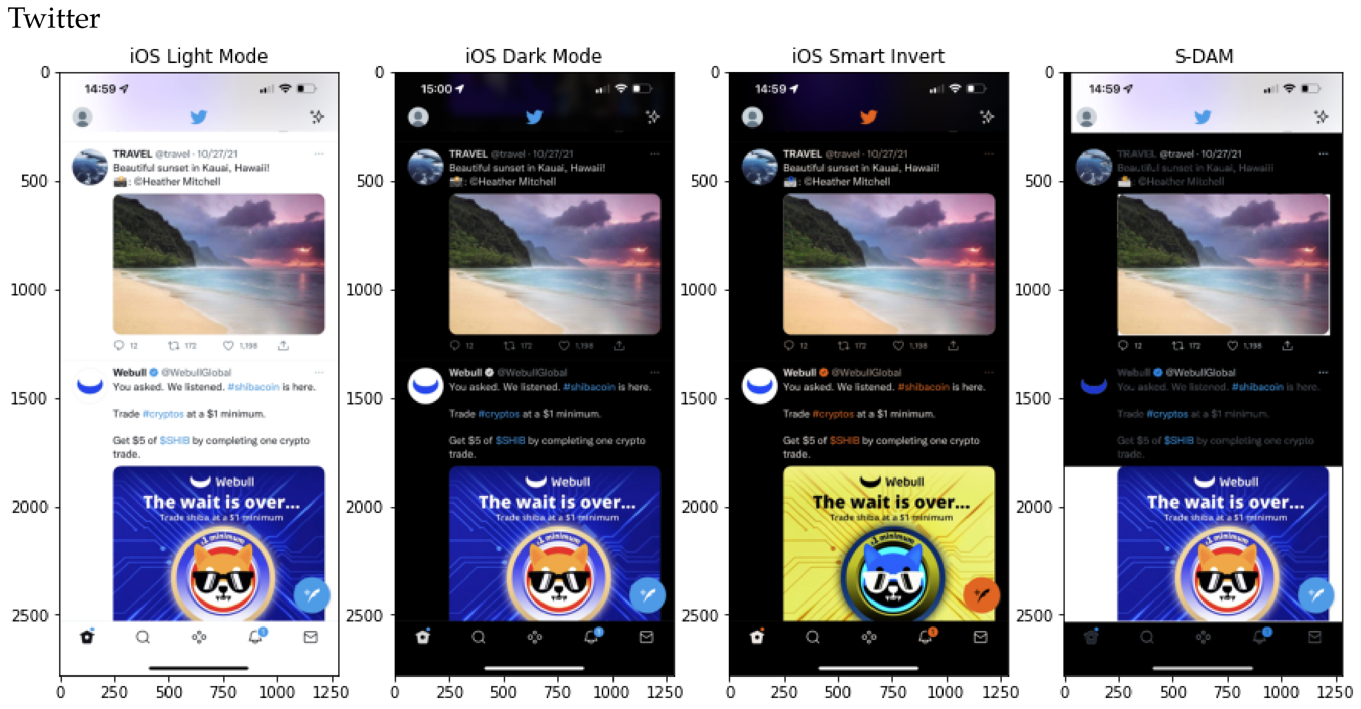

Twitter
iOS Light Mode
iOS Dark Mode
iOS Smart Invert
S-DAM
TRAVEL @travel · 10/27/21
Beautiful sunset in Kauai, Hawaii!
📸: ©Heather Mitchell
Webull @WebullGlobal
You asked. We listened. #shibacoin is here.
Trade #cryptos at a $1 minimum.
Get $5 of $SHIB by completing one crypto trade.
The wait is over...
Trade shiba at a $1 minimum

Figure 11: **A side by side view of screenshots from Twitter app in Light Mode, Smart Invert, and S-DAM.**

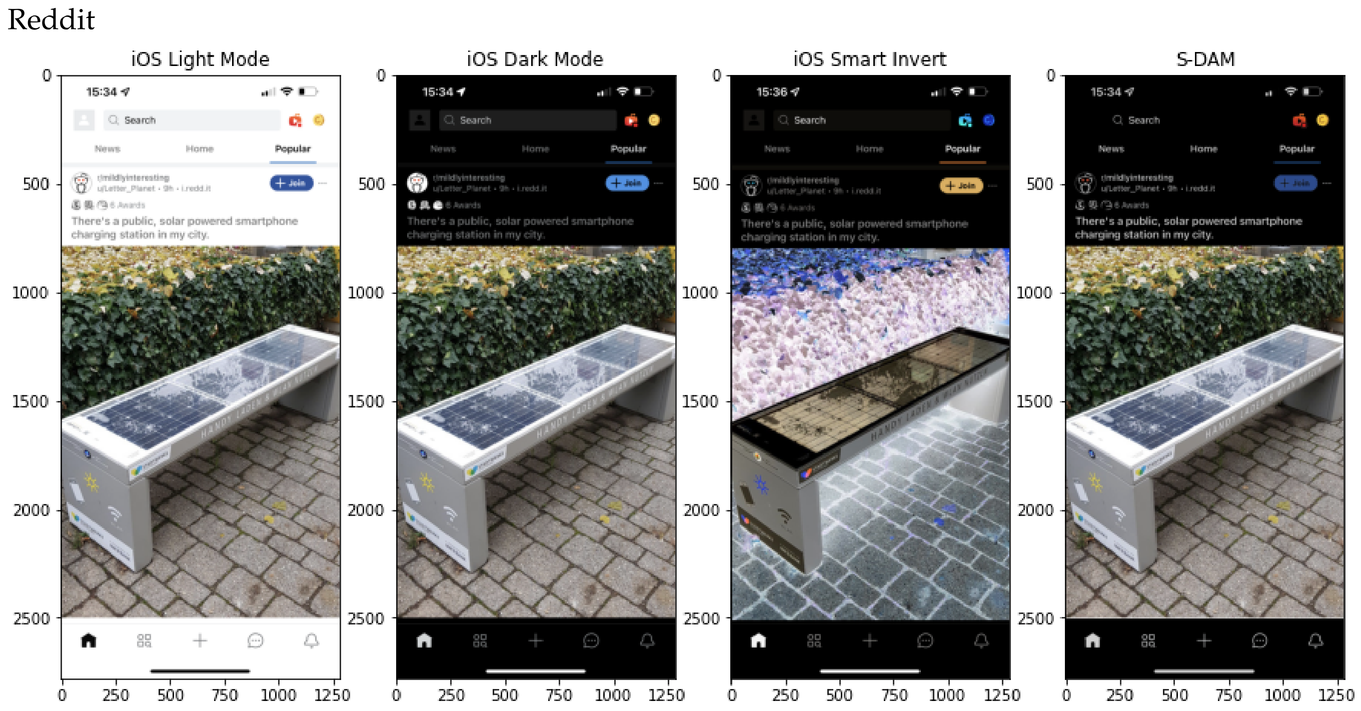
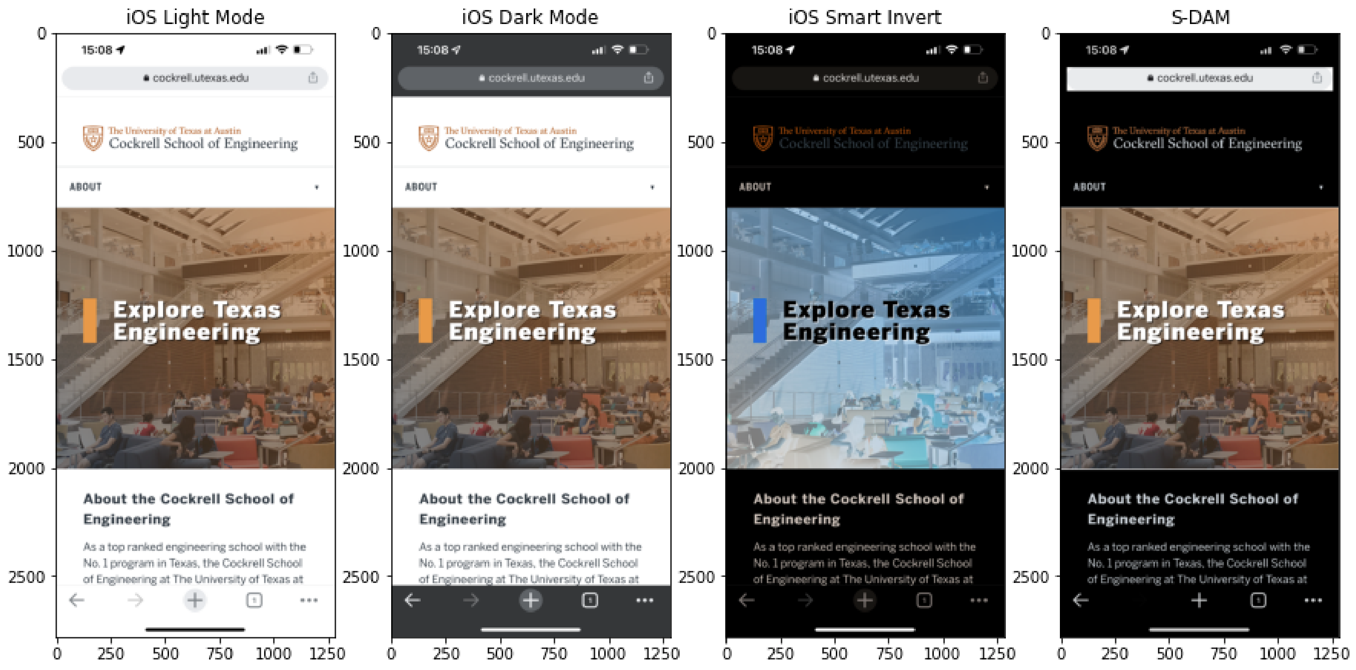


Figure 12: **A side by side view of screenshots from Reddit app in Light Mode, Smart Invert, and S-DAM.**

Chrome Browser



About the Cockrell School of Engineering

As a top ranked engineering school with the No. 1 program in Texas, the Cockrell School of Engineering at The University of Texas at

Figure 13: **A side by side view of screenshots from Chrome Browser app in Light Mode, Smart Invert, and S-DAM.**