Optical Character Recognition to Text

Prepared for
Anthony Aighobani
Kamloops, BC

Prepared by
Gustavo Kang Shim
Thompson Rivers University

November 26th, 2022

Table of Contents

Introduction

I am Gustavo Kang Shim. Currently a 3rd year student in Computer Science at Thompson Rivers University. I have done some Hackathons in the past, as well as many websites, apps and softwares. The project will be mostly conducted by me, and I do not currently have any agency nor contracts with no other than myself. I believe I'm suited for making such an app because of my given background in Computing Science in general.

Project Description

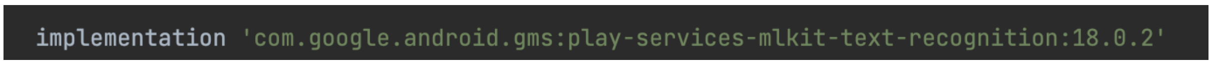
The app itself is basically a combination of existing ideas. The app first has a main activity where three buttons are located. Also it includes a background gradient of orangish, as well as a textView that will contain the text provided by the user that will be detected soon after. Once clicking on the first button "Capture", the user's camera is used. There, a user takes a picture of some text. Once that is done, the user then is brought to the first main API called Ucrop. Once done, click the checkmark and then go back to the main activity. After that, hit "Detect" to use the second main API called Text Recognition, where it processes the Image to Text. Lastly, you are able to copy that to your own clipboard.

Services Process and Timeline as of the time of this report

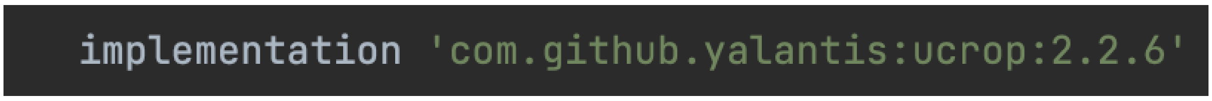
- Week 1: Scoping process and talking with interested testers for the project, asking for ideas from drivers around the area.
- Week 2: Get ideas started on the HOW of the picture.
- Week 3-9: Development of the prototype
- Week 10: Testing with Drivers
- Week 11: Working prototype is done, see if approved.

In-Depth of the Project

The two dependencies used were the following:



```
implementation 'com.google.android.gms:play-services-mlkit-text-recognition:18.0.2'
```



```
implementation 'com.github.yalantis:ucrop:2.2.6'
```

At first, I was worried about using TextRecognition, but it was a very easy API to use which allowed me more time to develop UCrop and Clipboard copy.All of the app was made using binding, which made it easier to access all of the items and things in the main activity file.

Refer below to the TextRecognition function:

```
private fun processImage(){

    if (imageBitmap !=null) { // if there is an image

        val image = imageBitmap?.let {
            InputImage.fromBitmap(it, 0) // assign image stored in bitmap here
        }
        image?.let {
            recognizer.process(it) // parse image to the textRecognition
                .addOnSuccessListener { visionText ->
                    binding.textView.text = visionText.text
                    Toast.makeText(applicationContext, "OCR-ed!",
Toast.LENGTH_SHORT).show()
                } // get text in the textview and toast success
                .addOnFailureListener { e ->
                    // if fail do nothing
                }
        }
    }
    else{
        Toast.makeText(this, "Please capture photo", Toast.LENGTH_SHORT).show()
      }
}
```

As for the clipboard, I used Clipboard manager. Please refer to the code below for the Clipboard function itself:

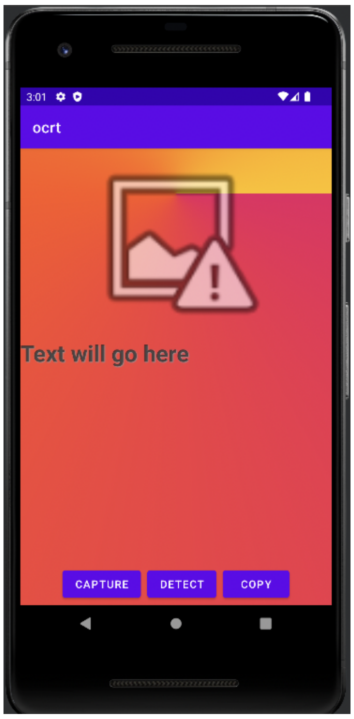
```
private fun Context.copyToClipboard(text: CharSequence){
        val clipboard =
ContextCompat.getSystemService(this,ClipboardManager::class.java) // get the
clipboard of system
        clipboard?.setPrimaryClip(ClipData.newPlainText("",text)) // set the text of
clipboard to the text parsed
    }
```

And lastly, this is what the Ucrop was doing. First, it launches the camera, then on the result it parses the image to Ucrop, lastly, it sets imageView to the image returned by the Ucrop

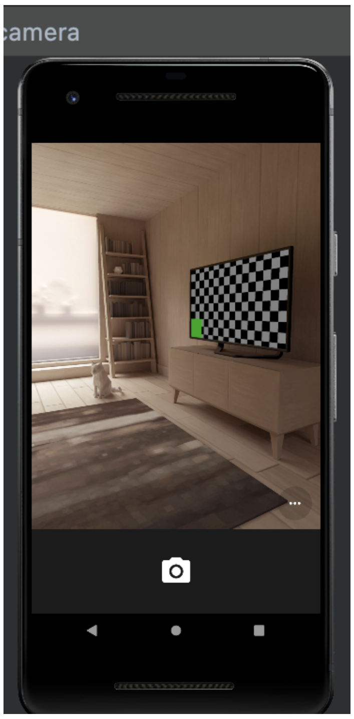
```
    }

private fun launchImageCrop(uri: Uri) {
        var destination:String=StringBuilder(UUID.randomUUID().toString()).toString()
        var options:UCrop.Options=UCrop.Options()
        UCrop.of(Uri.parse(uri.toString()), Uri.fromFile(File(cacheDir,destination)))
            .withOptions(options) // crop activity settings
            .withMaxResultSize(800, 800)
            .start(this)
    }
    private fun pickFromCamera(){
        val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        activityResultLauncher.launch(intent)
    }


    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (resultCode == RESULT_OK && requestCode == UCrop.REQUEST_CROP) {
            val resultUri : Uri?= UCrop.getOutput(data!!) // once the result is given from the
camera, get image in UCrop
            setImage(resultUri!!)
        }
    }
    private fun setImage(uri: Uri){
        imageBitmap = MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
        binding.imageView.setImageBitmap(imageBitmap) // set image in imageview
    }
```

- App

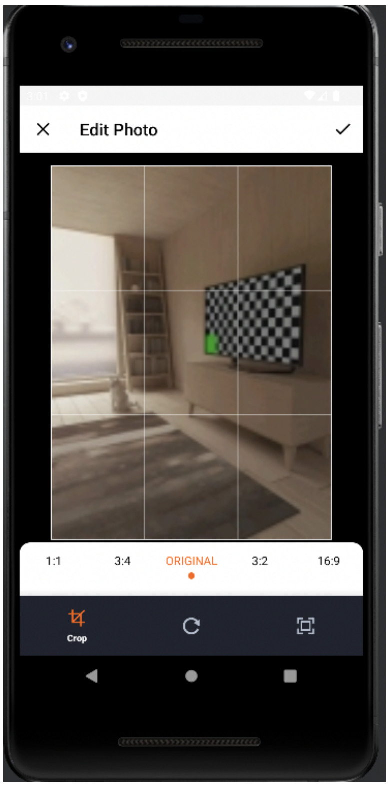
The app itself was really simple to build. The gradient was a drawable made with just some gradient colord (yellow, red, orange). As for the image in imageView as a fill, it was taken from the default android drawables.



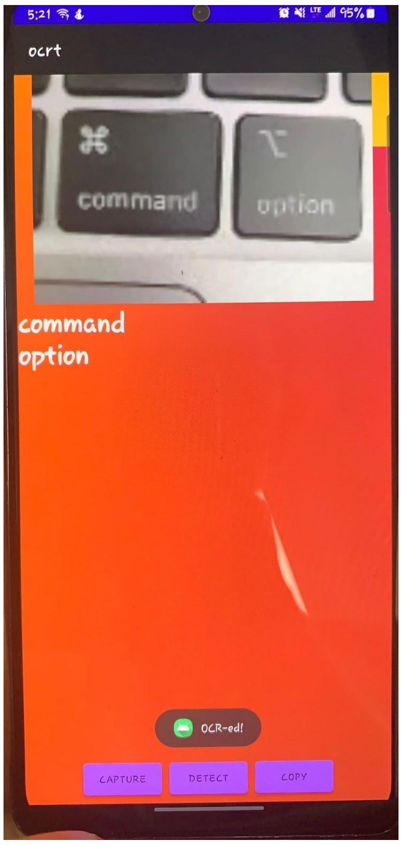
This is main activity, contains 3 buttons, one text view and a image view.



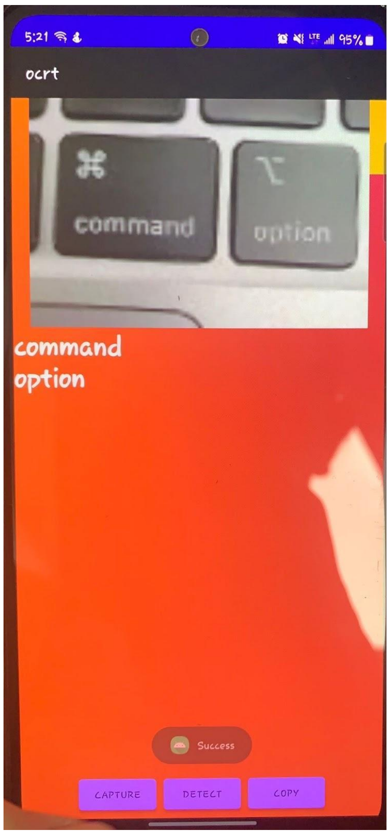
As capture is clicked, it bring the user to the camera.



Once done with the camera, the user is able to fully use the project’s Ucrop with it’s own UI.



As the image editing is done, it returns the imageView the image itself. Once the “Detect” button is clicked then it toasts OCR-ed and displays the result in textView.



Lastly, once clicked on copy button, it toasts a Success! Message to tell the user it has been copied to the clipboard.

- ## Budget of the project

Since the one person developing this project will be me, I myself will complete the project. The cost will be zero, as the app is not running on any servers, rather, it's totally client-run. Potentially in the future, a cost for maintenance of the app will arise. The app as of now will be free. For profit, Google Ads would be best.

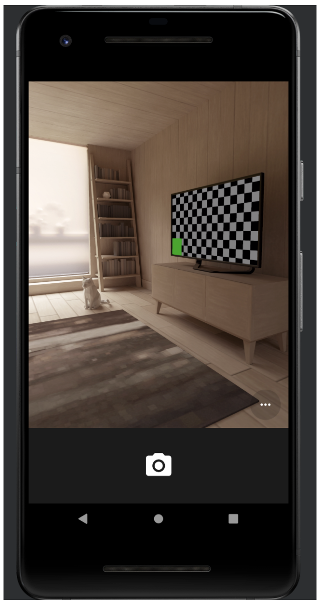
- ## Known Bugs and Issues

Bugs and issues were a big problem of this project as it had a lot of dependencies issues, as because of the mass deprecation of APIs, I ran into multiple problems while developing this app. As of now, bugs are minimal and are the following:

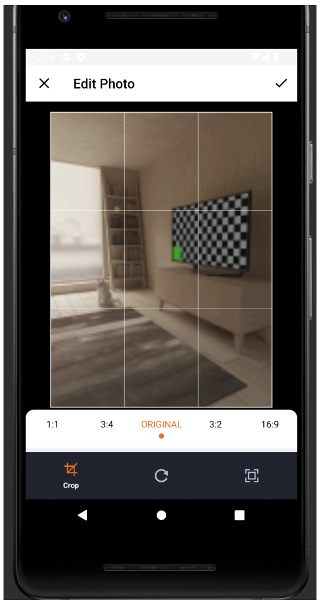
- No request for permission to use camera implemented
  - I tried implementing, but I was unable to fully test it. The function are made but unused.

```
 private fun checkPermission(): Boolean {
        return ContextCompat.checkSelfPermission(
            this,
            Manifest.permission.READ_EXTERNAL_STORAGE
        ) == PackageManager.PERMISSION_GRANTED &&
ContextCompat.checkSelfPermission(
            this,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
        ) == PackageManager.PERMISSION_GRANTED &&
ContextCompat.checkSelfPermission(
            this,
            Manifest.permission.CAMERA
        ) == PackageManager.PERMISSION_GRANTED
    }

    private fun requestPermission() {
        ActivityCompat.requestPermissions(
            this,
            arrayOf(
                Manifest.permission.READ_EXTERNAL_STORAGE,
                Manifest.permission.WRITE_EXTERNAL_STORAGE,
                Manifest.permission.CAMERA
            ),
            100
        )
    }
```

- Ucrop dependency bug makes the resolution worse.



—->



- ## Call To Action (Next Steps)

As of the last stage (Week11) , for the project to continue, an actual fully functioning App will need further enhancing and polishing on the UI. For us to continue, contact at gkangshim@gmail.com for further inquiries.

- Terms and Conditions

https://shimuraii.github.io/t-c_ma2/public.html please refer to the link.