Aplikasi Deteksi Nilai Uang pada Mata Uang Indonesia dengan Metode *Feature Matching*

Giovinna Khoharja[1], Liliana, M.Eng[2], Anita Nathania Purbowo, M.MT[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031)-2983455, Fax.(031)-8417658
giovinnakho@gmail.com[1], lilian@petra.ac.id[2], anitaforpetra@gmail.com[3]

ABSTRAK

Uang kertas tidak dapat dikenali dan dibedakan oleh tunanetra. Oleh karena itu pada tulisan ini, dibuat aplikasi untuk mengenali nilai uang kertas yang dapat berjalan pada smartphone Android dengan kamera, untuk membantu orang tersebut.

Rupiah Indonesia (IDR) digunakan sebagai data sampel. Aplikasi ini tidak membutuhkan komunikasi apapun dengan server dan semua komputasi yang diperlukan dilakukan pada handphone itu sendiri. Pada aplikasi ini, pengguna dapat mengambil gambar uang dengan melakukan double tap, kemudian hasilnya akan disuarakan. Sistem ini menggunakan algoritma visi komputer, seperti deteksi feature, deskripsi feature, dan matching. Pengaplikasian dari metode ORB, SURF, dan SIFT diterapkan untuk mencocokkan gambar yang diambil dan gambar template. Untuk memperbaik hasil dari pengenalan uang kertas, homography digunkana untuk menyaring hasil dari feature matching.

Sistem ini mengevaluasi performa dari 700 gambar dan mendapatkan hasil akurasi sebanyak 93.14% menggunakan SURF, 92.57% menggunakan SIFT, dan 89.17% menggunakan ORB.

Kata Kunci: Pengenalan mata uang, deteksi uang kertas, pencocokkan *feature*, deteksi nilai uang Indonesia, aplikasi untuk orang buta.

ABSTRACT

Banknote can't be detected and recognized by blind people. Application for banknote recognition, which can run on Android smartphone with camera, is made in this paper to help blind people.

Indonesian Rupiah(IDR) is used as a working example. This does not require any communication with remote server, and all the necessary computations take place on the phone itself. In this application, user can tak picture of the banknote with double tap on the screen, and the results will be announced by voice. The system relies on computer vision algorithm, such as feature detection, feature description, and matching. Each application of ORB, SURF, and SIFT is applied in matching captured banknote images to template images. To improve the confidence of the banknote recognition, homography is used to filter the feature matching results.

The systems evaluate the performance on 700 images and report an accuracy of 93.14% using SURF, 92.57% using SIFT, and 89.17% using ORB

Keywords: *Currency recognition, banknote recognition, feature matching, Indonesia currency detection, application for blind.*

1. INTRODUCTION

Uang mempunyai peranan penting dalam kehidupan sehari-hari, contohnya jual-beli, dan transaksi. Uang kertas Indonesia mempunyai motif yang berbeda-beda sehingga dapat dibedakan nilai uangnya dengan penglihatan saja. Menurut perkiraan Kementrian Kesehatan, 1.5% dari jumlah penduduk Indonesia mengalami kebutaan [18]. Teknologi *smartphone* saat ini memungkinkan penggunaan oleh tunanetra. Oleh karena itu, aplikasi deteksi nilai uang dapat menjadi nilai tambah bagi teknologi untuk tunanetra.

Dengan tujuan ini, diimplementasikan visi komputer yang dapat mendeteksi nilai uang dengan kamera. Pertama gambar kamera diambil untuk dieteksi *keypoint* dan *descriptor*-nya yang selanjutnya akan digunakan untuk menemukan kecocokkan yang paling baik dengan data *keypoint* dan *descriptor* pada basis data. Metode yang digunakan untuk mengambil *keypoint* dan *descriptor* adalah ORB (*Oriented FAST and Rotated BRIEF*), SURF (*Speeded-Up Robust Features*), dan SIFT (*Scale-Invariant Feature Transform*). Teknik yang digunakan untuk menyaring *feature* yang tidak sesuai, antara lain tes rasio [8], dan *homography*.

Ada beberapa masalah yang terjadi pada aplikasi ini. Yang pertama adalah semua komputasi dilakukan pada *handphone* itu sendiri, dimana pengkomputasian algoritma ORB, SURF, atau SIFT terhadap kumpulan basis data gambar *template* membutuhkan tenaga dan waktu yang banyak. Pengenalan uang kertas telah diteliti sebelumnya menggunakan berbagai macam metode, antara lain jaringan syaraf [1], *Local Binary Pattern* [16], *Multi-Layer Percepton* [3], *Backpropagation Neural Network* [15], *Radial Basis Function Network* [14]. Hampir seluruh penelitian tersebut memberikan hasil di atas 90% hingga 100% apabila penelitian menggunakan data sampel yang di-*scan* dan dimanipulasi sehingga seluruh detail uang terlihat jelas, dan tanpa rotasi.

Sedangkan masalah pada penelitian ini adalah kondisi pengambilan gambar tidak dapat ditentukan terlebih dahulu karena tunanetera tidak dapat menyadari lingkungan sekitarnya, dan pengmabilan gambar menggunakan kamera *handphone* sehingga tidak seluruh detail uang terlihat. Penelitian dengan data sampel dalam berbagai kondisi menggunakan metode antara lain, SURF [7], *face recognition* [9], *feature matching* dan RANSAC [4], SIFT dan ORB [17].

2. TINJAUAN PUSTAKA

Target pengguna aplikasi pendeteksi nilai uang merupakan orang buta atau orang yang memiliki masalah penglihatan berat. Oleh karena itu, pengguna tidak dapat menyadari kondisi kamera (pencahayaan, kontras, objek lain, *blur*), sehingga kemungkinan hasil foto memiliki skala dan rotasi yang bervariasi. Pengguna

memerlukan aplikasi sederhana yang tidak membutuhkan input apapun kecuali foto uang kertas [17] dan memberikan output berupa audio agar dapat dimengerti pengguna.

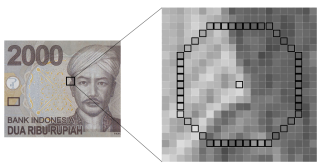
Pembuatan aplikasi skripsi ini menggunakan bahasa Java dengan program Android Studio. Bahasa Java merupakan bahasa umum yang sering digunakan dalam pembuatan aplikasi *mobile*. Android Studio digunakan karena memudahkan pembuatan *Graphical User Interface* (GUI) dan fungsi-fungsi dalam aplikasi *mobile*, karena sudah tersedia *class-class* bawahan dan simulasi antarmuka.

3. METODE

3.1 *Feature*

Feature adalah karakteristik yang unik dari suatu gambar. Deteksi *feature* merupakan hal pertama yang harus dilakukan dalam langkah pengenalan, dimana *feature* yang ditemukan berperan penting dalam pencocokkan dengan gambar *template*. Setelah itu dilakukan *feature description* yang memberi masing-masing *keypoint* yang ditemukan deskripsi atas area sekitarnya. Untuk menangani masalah kondisi pengambilan gambar, *descriptor* ini harus tahan terhadap perubahan rotasi dan skala. Algoritma yang dapat memenuhi hal tersebut adalah ORB, SURF, dan SIFT.

3.1.1 ORB



Gambar 1. Deteksi *feature* ORB dengan FAST-9

ORB [13] menggunakan metode FAST-9 (Gambar 1) untuk mendeteksi *feature*, dimana kandidat *keypoint* dibandingkan dengan pixel lingkaran sekitarnya dengan radius 9 pixel. Setiap *keypoint* akan dihitung massanya dengan:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{1}$$

Sehingga orientasi dari *keypoint* dapat dihitung menggunakan $\theta = atan2(m_{01}, m_{10})$. *Descriptor* dibuat dengan mengambil 31x31 *patch* di sekitar *keypoint.* Setiap *patch* dibuat 5x5 *subpatch*. ORB mengambil 2 *subpatch* untuk dievaluasi intensitasnya, apakah lebih besar *subpatch* pertama atau kedua sehingga mendapatkan nilai biner 0 dan 1. ORB melakukan *learning* untuk mendapatkan 256 string biner. Pertama mengurutkan *mean* pasangan *sub-patch* dari yang paling mendekati 0.5, dimana 0.5 menunjukkan variasi yang tinggi. Lalu, dilakukan *greedy search*: 1) memasukkan pasangan pertama ke vektor hasil R; 2) mengambil pasangan selanjutnya dan membandingkannya dengan seluruh pasangan pada vektor R dan apabila memiliki kolerasi yang tinggi maka akan dibuang; 3) melakukan langkah sebelumnya hingga vektor R beisi 256 pasangan.

3.1.2 SIFT

Algoritma SIFT [8] dirangkum menjadi 4 langkah berikut:

1) Deteksi ruang skala extrema : gambar akan diperkecil sebanyak oktaf σ, dan setiap oktaf diaplikasikan Gaussian *blur* dengan level *k.* DoG (*Difference of Gaussian*) diambil dari perbedaan gambar masing-masing oktaf.

2) Lokalisasi keypoint : setiap kandidat *keypoint* dicari maksimum dan minimumnya dengan membandingkan dengan 3x3x3 tetangga pada level yang sama, level diatas dan dibawahnya.

3) Penentuan orientasi kanonik : melakukan filter terhadap *keypoint* dengan kontras rendah dan *edge*. Hal ini dapat dilakukan dengan membuat Hessian matriks dari area *keypoint* dan diambil kedua nilai *eigenvalue*-nya dengan rumus berikut:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{2}$$

Dimana *trace* didapatkan dari $Tr(H) = D_{xx} + D_{yy}$ dan determinan didapatkan dari $Det(H) = D_{xx}D_{yy} - (D_{yy})^2$. Apabila lebih kecil daripada *threshold* 10, maka akan disimpan dan dihitung orientasi dan *gradient*-nya.

4) Deskripsi keypoint : 16x16 pixel dengan titik tengah *keypoint* dibagi menjadi 4x4 untuk dibuat histogramnya. Histogram masing-masing gradien berukuran 8-bin sehingga menghasilkan *descriptor* 128-dimensi [5].

3.1.3 SURF

SURF [2] mengandalkan gambar integral, *box filter*, dan Hessian matriks untuk mempercepat perhitungan [12]. Untuk menemukan *feature* digunakan determinan Hessian matriks berikut:

$$Det(\mathcal{H}_{approx}) = L_{xx}L_{yy} - (0.9L_{xy})^2 \tag{3}$$

Berbeda dengan SIFT yang memperkecil gambar terus menerus, SURF memperbesar kernel *box filter* untuk mempercepat mendapatkan *keypoint* yang *scale-invariant*. SURF menggunakan 3x3x3 *neighbor* dan metode pengeliminasian *feature* yang sama dengan metode SIFT. Haar *wavelet* berukuran 4σ diaplikasikan pada radius 6σ di sekitar *keypoint.* Hasilnya diberi bobot Gaussian dan direpresentasikan pada ruang vektor. Orientasi dominan, dipilih dengan merotasikan segmen lingkaran $\pi/3\frac{\pi}{3}$, dan vektor yang paling tinggi merupakan vektor yang orientasinya digunakan oleh *keypoint*. Haar *wavelet* juga digunakan untuk menghitung gradien. Setiap wilayah dibagi menjadi 4x4 sub-wilayah dan masing-masing sub-wilayah dibagi menjadi wilayah sampel 5x5 sehingga menghasilkan 64-dimensi vektor v:

$$v = \left\{\sum dx, \sum |dx|, \sum dy, \sum |dy|\right\} \tag{4}$$

3.2 *Feature Matching*

Feature matching melibatkan pendeteksian *feature* yang memiliki *descriptor* yang sama atau mirip. *Nearest neighbor* adalah metode *feature matching* yang dilakukan untuk mencari jarak terdekat antara *keypoint* dari gambar pertama dengan seluruh *keypoint* dari gambar kedua. Untuk mencari jarak antara *descriptor* a dan b digunakan kalkulasi *Euclidean distance.* Pengembangan dari metode ini adalah k-*nearest neighbor*, dimana setiap *descriptor* pada gambar pertama, dicari k-*descriptor* dengan jarak terdekat pada gambar kedua. Apabila k=2 dan mempunyai nilai hampir sama, maka *match* yang ditemukan kurang bagus. Dari tes rasio [8] ditemukan bahwa apabila perbandingan jarak kedua k lebih besar dari rasio 0.8, cenderung merupakan *incorrect matches.*

3.3 RANSAC

RANSAC (*Random Sample Consensus*) merupakan prosedur untuk mengestimasi transformasi yang dialami oleh sebuah objek atau gambar [6]. RANSAC digunakan untuk menyaring *false match* yang masih banyak ditemui pada hasil *feature matching*. Cara kerja RANSAC adalah sebagai berikut:

1) Memilih 4 pasangan *feature* secara *random*

2) Komputasi *homography* H dari keempat *feature* dengan DLT (*Direct Linear Transform*)

3) Proyeksi x ke x' dengan $x_i' = Hx_i$

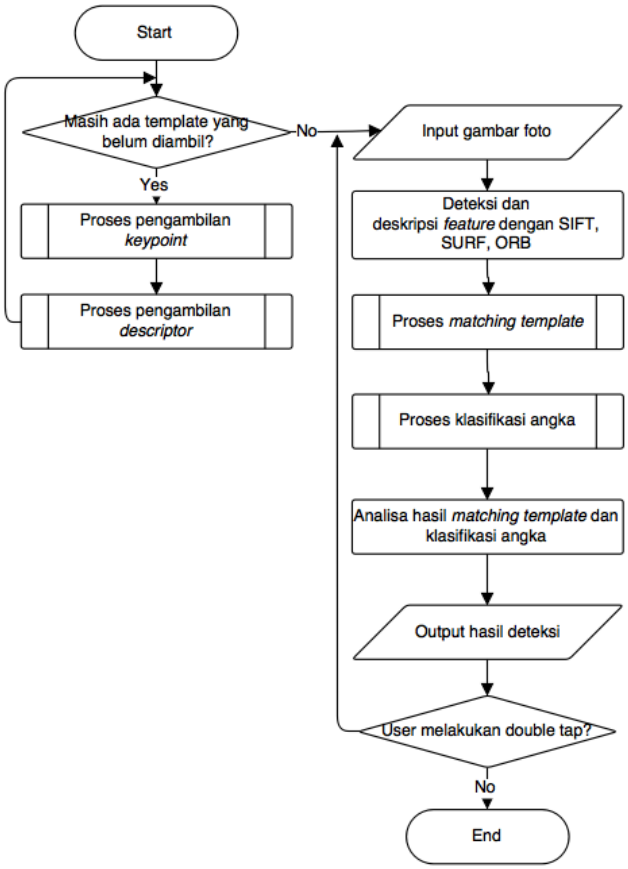
4) Hitung *match inlier* antara x dan x'dengan *distance* kurang dari *threshold.*

5) Ulangi langkah diatas hingga N-iterasi.

Setelah perulangan tersebut selesai, maka akan diambil *homography* H dengan *match inlier* paling banyak.

3.4 Penyimpanan *Feature Template*

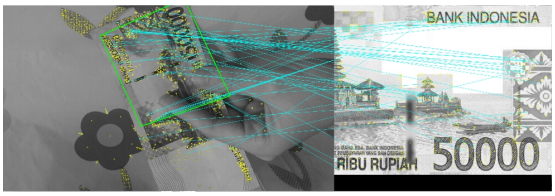
Penyimpanan *feature template* perlu dilakukan untuk mengurangi waktu komputasi dan menambah efisiensi. 28 gambar *template* dideteksi *feature*-nya menggunakan SIFT, SURF, atau ORB. *Keypoint* yang dihasilkan diubah bentuknya dalam bentuk JSON yang mempunyai properti data berikut *class_id*, x, y, *size*, *angle*, *octave*, dan *response*. *Descriptor* yang dihasilkan diubah bentuknya menjadi byte untuk ORB dan float untuk SIFT, dan SURF, karena *descriptor* ORB berbentuk biner sedangkan *descriptor* SIFT dan SURF 128-elemen dan 64-elemen. Data JSON tersebut akan disimpan dalam bentuk txt.



Gambar 2. *Flowchart* sistem kerja program

3.5 Sistem Kerja

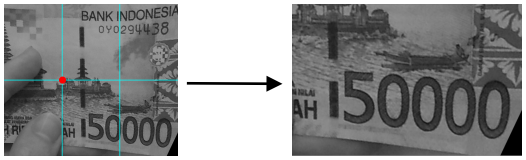
Sistem kerja aplikasi deteksi ini dapat dilihat pada Gambar 2, dimana pada saat awal program dijalankan, 28 file txt *keypoint* dan 28 file txt *descriptor* akan diambil dan diubah menjadi variabel yang dapat diproses oleh program. Ketika pengguna melakukan *double tap*, maka gambar akan diambil dan dilakukan deteksi *feature* (*input* gambar foto).



Gambar 3. Contoh hasil pengujian uang 50.000 rupiah menggunakan metode SURF

Pada tahap *matching template*, *feature* gambar foto tersebut akan dilakukan pencocokkan dengan *feature template*. Pencocokkan dilakukan menggunakan *2-nearest-neighbor* Hamming *distance* untuk ORB, dan *2-nearest-neighbor* Euclidian *distance* untuk SIFT dan SURF. Hasilnya akan dihitung rasionya, dimana yang lebih besar dari 0.8 akan dibuang. Apabila setelah tes rasio, hasil *match* yang didapatkan lebih besar dari 4, maka dilanjutkan menggunakan metode RANSAC, karena RANSAC membutuhkan minimal 4 *match random* untuk membentuk *homography*. Dari semua *template*, dicari kecocokkan dengan *inlier* yang paling tinggi, dan apabila lebih besar daripada *threshold* maka merupakan *template* yang cocok dengan gambar foto (Lihat Gambar 3).

Pada proses klasifikasi angka, dilakukan pemotongan bagian angka dari gambar foto. Proses ini membutuhkan *input* berupa matriks *homography* dan titik kotak perspektif yang didapatkan dari hasil klasifikasi. Langkah pertama yang dilakukan adalah dengan memotong dan memperspektifkan gambar foto sesuai dengan gambar *template*. Setelah itu, apabila pada *matching template* ditemukan bahwa *template* yang paling cocok adalah uang bagian depan kiri dengan nilai berapapun, maka gambar akan dipotong lagi dari titik (0,0) sepanjang ¾ bagian gambar dan setinggi ½ bagian gambar, karena gambar angka terdapat pada bagian tersebut. Apabila paling cocok dengan uang bagian belakang kanan dengan nilai berapapun, maka bagian uang akan dipotong dari titik (sepertiga *width*, ½ *height*) hingga ujung gambar seperti yang dapat dilihat pada Gambar 4. Hasil potongan tersebut akan dilakukan klasifikasi sama seperti metode sebelumnya, namun dengan *template* berbeda yaitu *template* angka sehingga didapatkan *template* angka mana yang paling cocok dengan gambar potongan.



Gambar 4. Memperspektifkan gambar foto pada Gambar 3, dan memotong bagian angka.

4. IMPLEMENTASI

Banyak aplikasi *mobile image retrieval* menggunakan internet, dimana *features* dikirim ke server untuk dicocokan dengan basis data online, sehingga tidak dibatasi seberapa besar basis data yang digunakan [10]. Namun, karena aplikasi pendeteksi nilai uang ini berbasis *offline,* maka harus dibatasi basis data yang digunakan [11]. Oleh karena itu, basis data hanya menyimpan file teks *feature* saja, yang rata-rata berukuran 200K. Audio untuk pengguna juga disimpan di dalam basis data dengan masing-masing maksimal berukuran 10K. Audio yang dibutuhkan adalah cara penggunaan saat aplikasi dijalankan, peletakkan kamera dan uang, cara mengambil gambar lagi, tujuh hasil nilai uang, dan pesan uang tidak terdeteksi.

Percobaan dilakukan terhadap 700 data yang terdiri dari masing-masing 50 gambar foto sisi depan dan sisi belakang 7 tipe uang

kertas Indonesia. Gambar foto tersebut berukuran 1980x1020 yang diambil diuji pada *smartphone* dengan kamera 13MP, Android 5.0.2 Lollipop, *CPU* : *Octa-core (4x1.7 GHz Cortex-A53 & 4x1.0 GHz Cortex-A53,* dan RAM 3G. Pengujian mengalami hambatan pada metode SIFT, dimana untuk deteksi 1500 *feature* pada 1 gambar membutuhkan waktu 24.559 detik dan 12.976 detik untuk 1000 *feature*. Oleh karena itu, untuk metode SIFT seluruh data sampel diubah menjadi ukuran 640x360. Pertama, dilakukan pengujian nilai parameter terbaik pada masing-masing metode. Hasil dari pengujian tersebut dapat dilihat pada Tabel 1.

Tabel 1. Hasil deteksi ORB, SURF, SIFT terhadap 700 data sampel

	ORB	SURF	SIFT
Keberhasilan(%)	89.714	93.142	92.571
Waktu(detik)	5.466	6.633	2.160
Inlier max gagal	57	20	46

Pengujian rotasi dilakukan dengan 14 foto uang kertas yang dirotasikan sebanyak kelipatan 30° sehingga setiap rotasi mempunyai 14 data sampel. Begitu juga dengan pengujian skala, 14 gambar uang kertas diperbesar dan diperkecil dari 10% hingga 150%. Selain itu, dilakukan pengujian terhadap coretan, dimana uang kertas 2000 rupiah diberi coretan dengan tebal 3px hingga 80px. Pada pengujian latar belakang, disiapkan 3 latar belakang yang mempunyai lebih dari 2 warna dan gambar uang akan ditempelkan pada latar belakang tersebut dengan besar 70% hingga 10% dari latar belakang. Pada pengujian potongan bagian uang, dengan memotong gambar uang hingga hanya terlihat 70%-10% bagian bergambar dari uang.

5. KESIMPULAN

Saat pengujian 700 data sampel, SURF merupakan metode dengan persentase keberhasilan terbaik saat mendeteksi yaitu 93.14%, diikuti dengan SIFT 92.571%, lalu ORB 89.714%. SIFT merupakan metode dengan waktu komputasi terlama apabila gambar berukuran 1980x1020 hingga dapat mencapai 12.976 detik untuk pendeteksian 1000 *feature,* dan 24 detik untuk pendeteksian 1500 *feature*. Sedangkan ORB merupakan metode dengan waktu komputasi tercepat yaitu 5.466 detik untuk pendeteksian 2000 *feature*, namun memiliki persentase keberhasilan terendah.

SIFT mengalami 100% keberhasilan pada pengujian rotasi dimana SURF dan ORB mengalami masing-masing 1 kegagalan. Seluruh metode memiliki rata-rata *keypoint* yang relatif lebih tinggi pada 0°, 90°, 180°, dan 270°. Pada derajat tersebut, total waktu deteksi SURF juga relatif lebih cepat dibandingkan derajat lainnya.

Pada pengujian skala ditemukan bahwa ORB merupakan metode yang paling tidak stabil saat pengujian skala, karena merupakan satu-satunya metode yang mengalami kegagalan saat skala diperbesar. SIFT berhasil mendeteksi foto dengan baik dan *inlier* di atas *threshold* hingga 20% dari ukuran 1920x1080, sedangkan SURF hingga 60%.

ORB mulai mengalami kegagalan pada pengujian coretan yang mempunyai besar 20px, SIFT 50px, dan SURF berhasil mendeteksi hingga coretan 80px. SURF memberikan hasil yang paling baik dalam pengujian latar belakang, dimana dapat mendeteksi dengan baik hingga gambar uang hanya 20% dari gambar latar belakang. SIFT masih dapat mendeteksi dengan baik hingga 40%, sedangkan ORB hingga 50%. SURF dan SIFT dalam mendeteksi potongan gambar dapat berhasil mendeteksi dengan baik hingga terlihat 60% bagian saja, sedangkan ORB hingga 70%.

6. REFERENSI

[1] Althafiri, E., Sarfraz, M., & Alfarras, M. (2012). Bahraini Paper Currency Recognition. *Design for Scientific Renaissance*, 104-115.

[2] Bay, H., Ess, A., Tuytelaars, T., & Gool, L. V. (2008, June). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding, 110*(3), 346-359. doi:10.1016/j.cviu.2007.09.014.

[3] Behjat, M., & Moallem, P. (2013). Fast and Low-cost Mechatronic Recognition System for Persian Banknotes. *International Journal of Advanced Robotic Systems*.

[4] Costa, C. M., Veiga, G., & Sousa, A. (2016). Recognition of Banknotes in Multiple Perspectives Using Selective Feature Matching and Shape Analysis. *IEEE International Conference on Autonomous Robot Systems and Competitions.* Bragança: IEEE.

[5] Do, T.-T., Kijak, E., Amsaleg, L., & Furon, T. (2012). Enlarging hacker's toolbox: deluding image recognition by attacking keypoint orientations. *ICASSP*, 1817-1820.

[6] Fischler, M. A., & Bolles, R. C. (1981, June). Random Sample Consensus: A Paradigm for Model Fitting with Applications To Image Analysis and Automated Cartography. *Communications of the ACM , 24*(6), 381-395.

[7] Hasanuzzaman, F. M., Yang, X., & Tian, Y. (2011). Robust and Effective Component-based Banknote Recognition by SURF Features. *IEEE Transactions on Systems Man and Cybernetics Part C*, 1-6.

[8] Lowe, D. G. (2004, November). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision, 60*(2), 91-110. doi:10.1023/B:VISI.0000029664.99615.94.

[9] Nojavani, I., Rezaeezade, A., & Monadjemi, A. (2014). *Iranian Cashes Recognition Using Mobile.* University of Isfahan, Computer Science & Information Technology, Isfahan.

[10] Panda, J., Brown, M. S., & Jawahar, C. V. (2013). Offline Mobile Instance Retrieval with a Small Memory Footprint. *Proceedings of the 2013 IEEE International Conference on Computer Vision* (pp. 1257-1264). Sydney: IEEE.

[11] Panda, J., Sharma, S., & Jawahar, C. V. (2012). Heritate App: Annotating Images on Mobile Phones. *Proceedings of the 8th Indian Conference on Vision, Graphics and Image Processing*, (pp. 16-19). Bombay

[12] Patel, A., Kasat, D., Jain, S., & Thakare, V. M. (2014, May). Performance Analysis of Various Feature Detector and Descriptor for Real-Time Video based Face Tracking. *International Journal of Computer Applications, 93*(1), 37-41. doi:10.5120/16183-5415.

[13] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2564-2571). Barcelona: IEEE.

[14] Sarfraz, M. (2015). An intelligent paper currency recognition system. *International Conference on Communication, Management and Information Technology* (pp. 538 – 545). Kuwait: Procedia Computer Science 65 .

[15] Sargano, A. B., Sarfraz, M., & Haq, N. (2014). An intelligent system for paper currency recognition with robust features. *Journal of Intelligent & Fuzzy Systems*, 1905–1913.

[16] Sharma, B., Kaur, A., & Vipan. (2012). Recognition of Indian Paper Currency based on LBP. *International Journal of Computer Applications , 59*, 24-27.

[17] Singh, S., Choudhury, S., Vishal, K., & Jawahar, C. V. (2014). Currency Recognition on Mobile Phones. *22nd International Conference on Pattern Recognition.* Stockholm.

[18] Suyopratomo. (2016, Januari 26). *Tunanetra Bukan Obyek*. Retrieved from Media Indonesia: http://www.mediaindonesia.com/index.php/news/read/26078/tunanetra-bukan-obyek/2016-01-26