Aplikasi Pengenalan Pola Batik Dengan Menggunakan Metode *Gray-Level Cooccurrence Matrix*

Ronald Kurniawan Tjondrowiguno[1], Rolly Intan[2], Kartika Gunadi[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
E-mail: rtjondrowiguno@gmail.com[1], rintan@petra.ac.id[2], kgunadi@petra.ac.id[3]

ABSTRAK

Indonesia adalah negara yang kaya budaya. Sejak 2 Oktober 2009 batik telah resmi diakui UNESCO sebagai warisan budaya asli Indonesia. Selain coraknya yang unik, batik memiliki makna filosofis yang mendalam. Meskipun begitu belum ada aplikasi yang dapat mengenalkan berbagai jenis kain batik kepada masyarakat. Diharapkan aplikasi ini dapat menjawab permasalahan tersebut.

Aplikasi ini menggunakan *gray-level cooccurrence matrix* untuk mengekstrak fitur tekstur dari sebuah gambar batik. Fitur tekstur yang dihasilkan membentuk suatu dataset yang dimanfaatkan untuk membentuk *decision tree*. Keputusan yang dibuat oleh *decision tree* adalah jenis *isen* atau isian yang terdapat pada gambar batik.

Hasil pengujian pengenalan batik tulis adalah akurasi maksimal yang dianggap rendah sebesar 47.62% saja. Penyebabnya diperkirakan adalah jenis batik tulis yang tidak memiliki pola tekstur.

Kata Kunci: Pengolahan Citra, *GLCM*, Kecerdasan Buatan, *Fuzzy Set, ID3, Decision Tree*

ABSTRACT

Indonesia is country with rich culture. Since October 2nd 2009 batik has been officially recognized by UNESCO as Indonesia's authentic cultural heritage. In addition to its unique patterns, batik has a deep philosophical significance. However there is no application that can introduce many kinds of batik to the society. This application is expected to address that issue.

This application uses gray-level cooccurrence matrix to extract texture features from an image of batik. The texture features extracted from a number of batik images create a dataset which can be used to create a decision tree. The decision made by the decision tree is the isen presented in batik image.

Test results of batik tulis recognition is maximum accuracy of 47.62%, which is considered low. The reason for that is supposedly the lack of texture patterns in batik tulis.

Keywords: *Image Processing, GLCM, Artificial Intelligence, Machine Learning, Fuzzy Set, ID3, Decision Tree*

1. PENDAHULUAN

Menurut [5], batik adalah kain bergambar yang pembuatannya secara khusus dengan menuliskan malam pada kain itu, kemudian pengolahannya diproses dengan cara tertentu. Kata "batik" sendiri berasal dari bahasa Jawa, yaitu *amba* yang berarti menulis dan *nitik* yang berarti titik. Pada tanggal 2 Oktober 2009, batik pertama kali diakui oleh UNESCO sebagai warisan budaya asli Indonesia [12]. Tetapi, masih banyak masyarakat Indonesia yang belum mengerti dan memahami filosofi dari berbagai pola batik. Salah satu contohnya adalah batik geringsing yang memiliki makna keseimbangan dalam hidup. Karena itu aplikasi ini memiliki nilai edukatif, yaitu dapat memberi informasi mengenai suatu pola *isen* batik kepada penggunanya.

Penelitian ini bertujuan membuat sebuah aplikasi untuk mengenali pola *isen* atau isian batik dengan menggunakan metode ekstraksi fitur *gray-level cooccurrence matrix*. Penulis memilih *gray-level cooccurrence matrix* sebagai metode ekstraksi fitur karena metode ini sudah pernah dipakai untuk mengenali tekstur dan memiliki akurasi hingga 80% pada [6].

Aplikasi ini dibuat menggunakan *platform desktop*, dengan bahasa pemrograman C#.NET. P*latform desktop* dipilih karena komputasi yang dilakukan oleh aplikasi ini kompleks sehingga membutuhkan kemampuan komputasi yang tinggi.

Diharapkan dengan adanya aplikasi ini, masyarakat Indonesia dapat lebih mengerti dan mengapresiasi batik sebagai warisan budaya asli Indonesia. Selanjutnya, diharapkan orang asing juga dapat dimudahkan untuk mempelajari batik sebagai budaya Indonesia.

2. TINJAUAN PUSTAKA

2.1 *Gray-level Cooccurrence Matrix*

Gray-level cooccurrence matrix adalah sebuah metode ekstraksi fitur yang digunakan untuk mendapatkan fitur *texture*/pola pada suatu citra digital. Cara kerja metode ini adalah dengan mencatat intensitas dua pixel yang berdekatan pada suatu gambar *gray-scale*. Ukuran dari GLCM mengikuti banyaknya level intensitas sebuah gambar. Pada gambar *grayscale* 8-bit, terdapat 2^8 = 256 level intensitas, sehingga ukuran GLCM untuk gambar *grayscale* 8-bit adalah 256x256. Untuk mengurangi beban komputasi, gambar *grayscale* 8-bit diubah menjadi gambar 4-bit saja. Hal ini bisa didapat dengan mengambil *4 most significant bits* dari gambar 8-bit, sehingga ukuran GLCM menjadi 16x16 saja. Penjelasan dan contoh pada bagian ini diambil dari [2].

GLCM menghitung seberapa sering pixel-pixel dengan *offset* jarak dan arah tertentu memiliki sebuah pasangan nilai. Sebagai contoh, Tabel 1 berisi gambar sederhana dengan 4 level intensitas.

Tabel 1. Contoh gambar sederhana

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

GLCM yang dihasilkan dengan *offset* (0,1) atau bersebelahan kiri-kanan dari Tabel 1 dapat dilihat pada Tabel 2.

GLCM menggunakan *zero-based indexing*. Untuk mendapatkan angka pada Tabel 2, setiap pixel yang berdekatan pada Tabel 1 dihitung jumlah kemunculannya. Sebagai contoh, angka 2 pada baris ke-0 kolom ke-1 adalah seberapa banyak intensitas 0 muncul di sebelah kiri dan intensitas 1 muncul di sebelah kanannya, angka 1 pada baris ke-0 kolom ke-2 adalah seberapa banyak intensitas 0 muncul di sebelah kiri dan intensitas 2 muncul di sebelah kanannya, dsb.

Tabel 2. Hasil GLCM dengan *offset* (0, 1)

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

Selanjutnya, GLCM yang dihasilkan dibentuk menjadi simetris terhadap sumbu diagonal utama. Hal ini dikarenakan kemunculan 2 pixel yang berdekatan bersifat dua arah (misal angka 0 di sebelah angka 2, maka angka 2 juga berada di sebelah angka 0). Proses ini dapat dilakukan dengan cara menjumlahkan GLCM dengan transposnya, sehingga GLCM menjadi seperti Tabel 3.

Tabel 3. Hasil GLCM yang simetris

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Pada tahap ini, nilai angka dalam GLCM bisa bervariasi sesuai ukuran gambar yang dihitung. Gambar yang besar akan memiliki jumlah elemen GLCM yang besar. Agar dapat digunakan lebih lanjut, hasil perhitungan GLCM harus dinormalisasi agar jumlah seluruh elemen GLCM menjadi 1. Normalisasi dilakukan dengan cara membagi seluruh isi GLCM dengan total jumlah semua isi GLCM seperti rumus pada (1). GLCM yang sudah dinormalkan menyatakan prosentase suatu elemen terhadap jumlah keseluruhan. Contoh hasil GLCM yang sudah dinormalisasi dapat dilihat pada Tabel 4.

$$p_{i,j} = \frac{v_{i,j}}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} v_{i,j}} \quad (1)$$

Secara formal, $p_{i,j}$ menyatakan elemen GLCM yang sudah dinormalisasi, $v_{i,j}$ menyatakan elemen GLCM yang belum dinormalisasi, dan N menyatakan ukuran dari GLCM.

Tabel 4 Hasil GLCM yang sudah dinormalisasi

0.167	0.089	0.044	0
0.089	0.167	0	0
0.044	0	0.250	0.044
0	0	0.044	0.089

GLCM yang sudah dinormalisasi ini dapat digunakan untuk mendapatkan kalkulasi yang akan dibahas pada subbab berikutnya.

Sebuah *offset* terdiri dari 2 bagian, yaitu arah (θ) dan jarak (d). Menurut [7], jarak d=1 ternyata memiliki hasil yang lebih baik daripada jarak d=2. Ada 4 arah *offset* yang umum dipakai, yaitu (0, 1), (1, 1), (0, 1), dan (-1, 1). Keempat *offset* ini masing-masing menggambarkan pixel yang bersebelahan dengan sudut 0^o, 45^o, 90^o, dan 135^o. Sudut 180^o, 225^o, 270^o, dan 315^o tidak dipakai karena sama dengan keempat *offset* umum tersebut.

GLCM sudah pernah digunakan dalam berbagai penelitian sebagai metode untuk mengevaluasi tekstur. Berbagai penelitian tersebut mencapai akurasi yang tinggi, yaitu 80% pada [6], 91,57% pada [8] dan 98.927% pada [10].

2.1.1 *Contrast*

Nilai *contrast* menggambarkan seberapa besar perbedaan intensitas antar *pixel-pixel* yang berdekatan. Semakin kontras perbedaan antara *pixel-pixel* yang berdekatan, semakin tinggi nilai *contrast*. Nilai *contrast* merupakan penghitungan derajat kedua. Rumus perhitungan *contrast* dapat dilihat pada (2).

$$Contrast = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} p_{i,j}(i-j)^2 \quad (2)$$

2.1.2 *Homogeneity (Inverse Difference Moment)*

Nilai *homogeneity* menggambarkan seberapa homogen intensitas antar pixel-pixel yang berdekatan. Semakin kontras perbedaan antara pixel-pixel yang berdekatan, semakin rendah nilai *homogeneity*. Nilai *homogeneity* merupakan penghitungan derajat kedua. Rumus perhitungan *homogeneity* dapat dilihat pada (3).

$$Homogeneity = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \frac{p_{i,j}}{1+(i-j)^2} \quad (3)$$

2.1.3 *Correlation*

Nilai *correlation* menggambarkan dependensi / korelasi antar pixel-pixel yang berdekatan. Nilai *correlation* merupakan perhitungan derajat kedua. Rumus perhitungan *correlation* dapat dilihat pada (4).

$$Correlation = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} p_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right] \quad (4)$$

2.1.4 *Energy*

Nilai *energy* atau disebut juga *entropy* menggambarkan perubahan nilai intensitas dalam suatu gambar. Istilah *entropy* diambil dari istilah fisika termodinamika, yaitu banyaknya energi yang hilang menjadi panas. Rumus perhitungan *energy* dapat dilihat pada (5).

$$Energy = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} {p_{i,j}}^2 \quad (5)$$

2.2 *Decision Tree*

Decision tree adalah struktur *tree* yang menyerupai *flowchart*, dimana tiap *internal node* menandakan pengetesan terhadap suatu atribut, tiap cabang menyatakan hasil dari tes tersebut, dan *leaf node* menyatakan sebuah kelas-kelas atau distribusinya menurut [4]. Penggunaan *decision tree* adalah untuk menghasilkan dan memvisualisasikan *rule-rule* yang dihasilkan dari data, sehingga membuat hasil lebih mudah dipahami dan akurat [11].

Dalam *decision tree*, tiap *node leaf* memiliki salah satu keputusan dan tiap *node branch* memiliki sebuah pilihan. Dimulai dari *root*, data akan diolah untuk menentukan pilihan di masing-masing *node*. Jalur yang diambil berbeda-beda, tergantung pilihan apa yang dipilih oleh data pada *node branch*. Data terus diuji dan pindah ke *node* selanjutnya hingga data mencapai suatu *node leaf*. Maka dapat dikatakan bahwa data itu akan menghasilkan keputusan sesuai *node leaf* yang dicapai data.

2.2.1 *ID3*

ID3 adalah algoritma yang dapat digunakan untuk menghasilkan *decision tree* dari sebuah *dataset* [1]. ID3 (*Iterative Dichotomizer* 3) dikembangkan untuk menentukan dari pola saja apakah sebuah posisi tertentu dalam pertandingan catur antara raja dan benteng melawan raja dan kuda akan berakhir dengan kalahnya raja dan kuda [9]. Dalam perkembangannya, algoritma ID3 cocok untuk pengambilan keputusan dengan dataset yang bersifat diskrit.

Atribut-atribut dataset yang digunakan dalam algoritma ID3 terdiri dari 2 macam, yaitu atribut penyebab (*antecedent*) dan atribut akibat (*consequent*). Atribut penyebab adalah atribut yang digunakan untuk menentukan nilai atribut akibat. Algoritma ini menggunakan kalkulasi entropi dalam data untuk menentukan atribut mana yang paling menentukan suatu kolom keputusan. Entropi adalah ukuran seberapa sebuah dataset itu tersebar (*random*). Nilai entropi dihitung sebagai relasi antara sebuah atribut penyebab dengan atribut akibat. Atribut dengan nilai entropi terrendah menandakan bahwa atribut tersebut paling menentukan atribut akibatnya.

Setiap atribut *consequent* memiliki *information need*, yaitu ukuran seberapa besar informasi yang dibutuhkan untuk dapat menentukan atribut *consequent* tersebut.

Dataset S didefinisikan sebagai berikut. Misal *m* adalah banyaknya macam nilai pada atribut *consequent* (sehingga ada *m* kelas pada atribut *consequent*), *s* adalah ukuran dataset, dan s_i adalah banyaknya data yang atribut *consequent*-nya bernilai c_i. Rumus untuk menghitung *information need* dapat dilihat pada (6) dan (7).

$$I(s_1, s_2, \dots, s_m) = -\sum_{i=1}^{m} p_i . log_2(p_i) \quad (6)$$

$$p_i = \frac{s_i}{s} \quad (7)$$

Misal suatu atribut *antecedent* mempunyai *v* macam nilai yang berbeda. s_{ij} didefinikan sebagai banyaknya data dengan atribut *antecedent* bernilai s_j dan atribut *consequent* bernilai c_i. *Entropy* dapat dihitung menggunakan (8), (9), dan (10).

$$E(A) = \sum_{j=1}^{v} \frac{s_j}{s} I(s_{1j}, s_{2j}, \dots, s_{mj}) \quad (8)$$

$$I(s_{1j}, s_{2j}, \dots, s_{mj}) = -\sum_{i=1}^{m} p_{ij} . log_2(p_{ij}) \quad (9)$$

$$p_{ij} = \frac{s_{ij}}{s_j} \quad (10)$$

Jika *entropy* mengukur persebaran nilai dalam sebuah *dataset*, kebalikannya adalah *information gain*. *Information gain* dapat dihitung menggunakan (11).

$$Gain(A) = I(s_1, s_2, \dots, s_m) - E(A) \quad (11)$$

2.2.2 *Fuzzy Decision Tree*

Penjelasan pada bagian ini diambil dari [3]. *Fuzzy Decision Tree* adalah pengembangan dari *decision tree induction* yang dikombinasikan dengan sistem logika *fuzzy*. Pada *fuzzy decision tree*, *rule* yang dibuat menggunakan *fuzzy set* sebagai sebab (*antecedent*) dan akibat (*consequent*). Pada *decision tree* klasik percabangan pada nilai numerik membutuhkan adanya titik potong.

Sebagai contoh, sebuah sistem menggunakan data suhu untuk menentukan kecepatan kipas angin. Dalam teori *decision tree* klasik, *rule* yang mungkin terbentuk adalah 'Jika suhu > 25^{o}C maka kecepatan = 20 putaran/detik'. Kelema *decision tree* klasik adalah jika suhu yang didapat adalah 24.9^{o}C maka rule tersebut tidak dijalankan. Dalam *fuzzy decision tree*, *rule* yang mungkin terbentuk adalah 'Jika suhu = panas maka kecepatan = tinggi', dengan *fuzzy set* panas untuk himpunan suhu udara dan *fuzzy set* tinggi untuk himpunan kecepatan putaran kipas.

Misal S menyatakan himpunan yang terdiri dari *s* sampel data. Himpunan atribut memiliki *m* nilai, yaitu v_i (untuk i=1, …, m), membentuk *m* kelas yang berbeda, C_i (untuk i=1, …, m). Terdapat pula *n fuzzy label*, F_j (untuk j=1, …, n) yang didefinisikan pada *m* atribut, v_i. $F_j(v_i)$ menyatakan derajat keanggotaan v_i pada *fuzzy set* F_j. Misal β_j adalah weighted sample pada fuzzy set F_j yang dapat dihitung seperti pada (12). Informasi yang dibutuhkan untuk mengklasifikasi didapat dari (13) dan (14).

$$\beta_j = \sum_i^m \det(C_i) \times F_j(v_i) \quad (12)$$

$$I(\beta_1, \beta_2, \dots, \beta_n) = -\sum_{j=1}^{n} p_j log_2(p_j) \quad (13)$$

$$p_j = \frac{\beta_j}{s} \quad (14)$$

Misal atribut *A* memiliki *u* nilai yang berbeda, $\{a_1, a_2, \dots, a_u\}$, yang mendefinisikan *u* kelas yang berbeda, B_h (untuk h=1, …, u). Misal ada *r fuzzy label*, T_k (untuk k=1, …, r) yang terdefinisi pada *A* sehingga T_k memenuhi (15).

$$\sum_k^r T_k(a_h) = 1 , \forall h \in \{1, \dots, u\} \quad (15)$$

Nilai *entropy* dari atribut *A* dapat dihitung menggunakan (16), (17), dan (18), sehingga nilai *information gain* juga dapat dihitung menggunakan (19).

$$E(A) = \sum_{k=1}^{r} \frac{\alpha_{1k} + \dots + \alpha_{nk}}{s} I(\alpha_{1k}, \dots, \alpha_{nk}) \quad (16)$$

$$\alpha_{jk} = \sum_h^u \sum_i^m \min(F_j(v_i), T_k(a_h)) \times \det(C_i \cap B_h) \quad (17)$$

$$I(\alpha_{1k}, \dots, \alpha_{nk}) = -\sum_{j=1}^{n} p_{jk} log_2(p_{jk}) \quad (18)$$

$$Gain(A) = I(\beta_1, \beta_2, \dots, \beta_n) - E(A) \quad (19)$$

3. ANALISIS DAN DESAIN

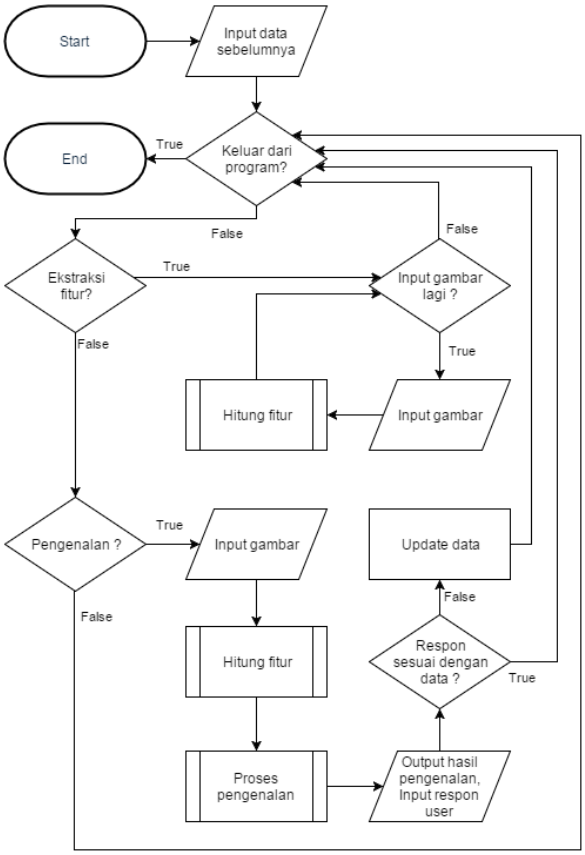
3.1 Arsitektur Sistem

Sistem yang dibuat terdiri dari 2 bagian yang berbeda. Bagian pertama digunakan untuk mengolah gambar batik menggunakan metode *GLCM* dan menghasilkan fitur. Data fitur dari bagian pertama dianalisa untuk menghasilkan *decision tree*. *Decision tree* kemudian diaplikasikan pada bagian kedua, yaitu sistem pengenalan untuk menentukan jenis dari sebuah gambar batik. Diagram alir sistem dapat dilihat pada Gambar 1.

Bagian yang pertama digunakan untuk melakukan proses ekstraksi fitur. Fitur-fitur yang merupakan hasil dari proses ini disimpan ke dalam suatu tabel. Selanjutnya fitur-fitur pada tabel ini dianalisa untuk menghasilkan *decision tree* yang digunakan pada bagian kedua untuk proses pengenalan gambar batik. Gambar batik yang digunakan untuk pembuatan *decision tree* dan proses pengenalan dibedakan menjadi 2 *dataset*, yaitu *learning dataset* untuk pembuatan *decision tree* dan *testing dataset* untuk proses pengenalan.

Fitur yang didapatkan dari gambar batik merupakan hasil dari kalkulasi GLCM. Kalkulasi yang digunakan adalah Contrast, Correlation, Homogeneity, dan Energy. Ukuran GLCM yang

digunakan adalah 8x8. Offset GLCM yang digunakan ada 4, yaitu horizontal ([0, 1]), vertikal ([-1, 0]), serta kedua diagonal ([-1, -1] dan [-1, 1]). Untuk masing-masing offset dibuat sebuah GLCM dan untuk masing-masing GLCM dihitung 4 kalkulasi, sehingga secara keseluruhan ada 16 fitur GLCM untuk tiap gambar batik.



Gambar 1. Garis besar sistem

Fitur yang sudah dihitung digunakan untuk membuat *decision tree* menggunakan algoritma ID3. Algoritma ID3 yang digunakan telah dimodifikasi, yaitu menggunakan *fuzzy set* dan *fuzzy value* untuk menghitung kebutuhan informasi (*information need*) dari data dan *information gain* untuk masing-masing fitur. *Information gain* digunakan untuk menentukan fitur mana yang digunakan pada setiap *node*. Karena fitur berupa bilangan *real*, maka digunakan titik potong untuk membagi data menjadi 2 kelompok.

3.2 Desain *Class*

Terdapat 2 *class* yang digunakan dalam pembuatan aplikasi ini, yaitu sebuah *static class* untuk pengolahan gambar batik menggunakan GLCM dan sebuah *class fuzzy set*. Untuk memberi penjelasan masing-masing *class*, subbab ini akan dibagi menjadi 2 subbab, masing-masing untuk *class* GLCM dan *class* FuzzySet.

3.2.1 *Desain Class GLCM*

Seperti sudah disebutkan pada bagian sebelumnya, *class* GLCM adalah sebuah *static class*. Hal ini dikarenakan class GLCM sendiri tidak perlu menyimpan data dari gambar inputnya maupun data hasil penghitungan GLCM. Data yang disimpan berupa daftar *offset* yang digunakan untuk pembuatan GLCM. *Offset* yang digunakan dapat dikustomisasi, namun untuk penelitian ini *offset* yang digunakan sesuai dengan subbab 3.1. Struktur *class* GLCM dapat dilihat pada Gambar 2.

GLCM
static int _offsetSize; static Tuple<int, int>[] _offsets;
static GLCM() static Image<Gray, Byte> Scale() static double[,,] CreateGLCM() static double[] Correlation() static double[] Contrast() static double[] Energy() static double[] Homogeneity() static double[] RowMean() static double[] ColMean() static double[] RowStdDev() static double[] ColStdDev()

Gambar 2. Class GLCM

Keterangan fungsi:

1. GLCM() adalah constructor untuk class GLCM. Prosedur ini berfungsi untuk mengisi variabel _offsetSize dan _offsets.
2. Scale() adalah fungsi untuk mengubah nilai dalam gambar agar dapat ditampung dalam GLCM.
3. CreateGLCM() adalah fungsi untuk membuat GLCM.
4. Correlation() adalah fungsi untuk menghitung *correlation* dari GLCM.
5. Contrast() adalah fungsi untuk menghitung *contrast* dari GLCM.
6. Energy() adalah fungsi untuk menghitung *energy* dari GLCM.
7. Homogeneity() adalah fungsi untuk menghitung *homogeneity* dari GLCM.
8. RowMean() adalah fungsi untuk menghitung *weighted mean* pada GLCM dengan *weight* berupa indeks baris.
9. ColMean() adalah fungsi untuk menghitung *weighted mean* pada GLCM dengan weight berupa indeks kolom.
10. RowStdDev() adalah fungsi untuk menghitung standar deviasi dari GLCM dengan menggunakan indeks baris sebagai *weight*.
11. ColStdDev() adalah fungsi untuk menghitung standar deviasi dari GLCM dengan menggunakan indeks kolom sebagai *weight*.

4. HASIL

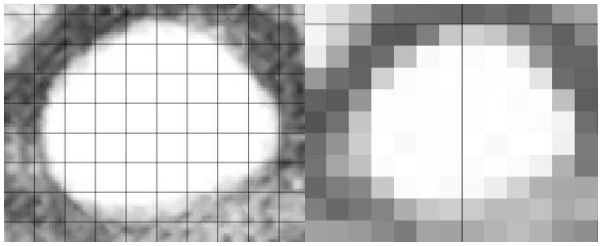
4.1 Pengujian Akurasi Pengenalan

Pada subbab ini akan diuji seberapa akurat proses pengenalan menggunakan *Fuzzy* ID3. Pengujian ini dilakukan untuk menguji kemampuan sistem dalam menebak jenis dari input gambar batik. Akurasi didefinisikan sebagai perbandingan jumlah gambar yang berhasil ditebak dengan jumlah gambar dalam *testing dataset* seperti dalam (20). Pada dataset dengan jumlah data yang sama, semakin tinggi jumlah tebakan yang benar maka nilai akurasi menjadi semakin tinggi.

$$Akurasi = \frac{jumlah\_tebakan\_benar}{jumlah\_data} \quad (20)$$

4.1.1 *Pengujian Dengan Dataset Batik Tulis*

Proses pengenalan akan diuji tingkat akurasinya dengan menggunakan *dataset* batik tulis. Untuk pengujian ini terdapat 2 parameter yaitu resolusi gambar dan ukuran GLCM yang dihasilkan. Resolusi gambar diperkirakan dapat mempengaruhi hasil pengujian karena resolusi gambar yang lebih besar memiliki jarak antar objek pada gambar yang lebih jauh. Diperkirakan semakin kecil resolusi gambar yang digunakan semakin akurat akurasi yang dihasilkan. Ada 4 macam resolusi gambar yang diuji, yaitu 1024x576, 512x288, 256x144, dan 128x72. Ilustrasi gambar yang sama dengan resolusi yang berbeda dapat dilihat pada Gambar 3.



Gambar 3. Ilustrasi gambar yang sama beda resolusi

Ukuran GLCM yang berbeda diperkirakan dapat mempengaruhi hasil pengujian karena ukuran GLCM yang berbeda untuk gambar yang sama akan menghasilkan fitur GLCM yang berbeda. Ada 3 macam ukuran GLCM yang digunakan, yaitu 4x4, 8x8, dan 16x16. Hasil pengujian secara keseluruhan dapat dilihat pada Tabel 5.

Tabel 5 Akurasi hasil pengujian batik tulis

Resolusi\GLCM	4x4	8x8	16x16
1024x576	26.19%	21.43%	30.95%
512x288	40.48%	35.71%	47.62%
256x144	40.48%	42.86%	40.48%
128x72	35.71%	47.62%	42.86%

Dari Tabel 5 dapat dilihat bahwa secara keseluruhan hasil pengenalan masih kurang memuaskan, dengan akurasi maksimal hanya 47.62%. Ada beberapa faktor yang diperkirakan membuat hasil kurang maksimal, misal faktor adanya bagian dari gambar yang bukan merupakan *isen* batik. Pada kenyataannya tidak mungkin sebuah kain batik hanya terdiri dari *isen* saja karena *isen* pada batik digunakan sebagai latar belakang saja. Namun perlu dicoba pengenalan dengan menggunakan gambar *isen* saja.

4.1.2 *Pengujian Dengan Tambahan Gambar Isen*

Salah satu hal yang diperkirakan menyebabkan akurasi dari pengenalan menjadi rendah adalah karena pada gambar batik terdapat banyak ornamen-ornamen batik yang bukan merupakan *isen* batik yang berpola. Pada subbab ini akan dilakukan pengujian dengan menggunakan gambar-gambar yang berisi *isen* saja untuk membantu *training dataset*. *Dataset* yang digunakan sama dengan pengujian pada Subbab 5.3.1, dengan ditambahkan gambar yang berisi *isen* saja pada *training dataset*. Ukuran gambar yang berisi *isen* saja bervariasi tergantung daerah *isen* pada gambar batik, namun resolusi untuk tiap ukuran batik tetap dipertahankan. Hal ini penting agar resolusi gambar batik secara keseluruhan tetap konsisten. Hasil pengujian dapat dilihat pada Tabel 6.

Tabel 6 Akurasi hasil pengujian dengan *isen*

Resolusi\GLCM	4x4	8x8	16x16
1024x576	33.33%	16.67%	21.43%
512x288	35.71%	33.33%	40.48%
256x144	40.48%	35.71%	47.62%
128x72	28.57%	47.62%	38.10%

Seperti dapat dilihat pada Tabel 6, hasil pengujian dengan tambahan input berupa gambar *isen* saja ternyata hampir sama dengan hasil pengujian tanpa tambahan input berupa gambar *isen* saja. Akurasi paling besar pada hasil pengujian ini juga sama dengan pengujian sebelumnya, yaitu 47.62%. Dari hasil ini dapat disimpulkan bahwa penambahan input berupa gambar *isen* saja ternyata tidak dapat meningkatkan akurasi pengenalan.

4.1.3 *Pengujian Dengan Dataset Batik Cetak*

Dari rendahnya akurasi pada pengujian menggunakan *dataset* batik tulis ada kemungkinan hal ini dikarenakan batik tulis yang bersifat *handmade* tidak memiliki pola berupa tekstur. Untuk menguji hal tersebut maka diperlukan pengujian dengan menggunakan *dataset* lain, yaitu *dataset* batik cetak. Jika pernyataan diatas benar, maka hasil pengujian pada subbab ini lebih baik daripada pengujian pada subbab sebelumnya. Pada pengujian ini digunakan 1 parameter saja, yaitu ukuran GLCM. Sama seperti pada subbab sebelumnya, ada 3 ukuran GLCM yang digunakan yaitu 4x4, 8x8, dan 16x16. Hasil pengujian dapat dilihat pada Tabel 7.

Seperti dapat dilihat pada Tabel 7, pengujian dengan dataset batik cetak menghasilkan akurasi terbesar 72.86%. Dapat dilihat juga bahwa secara keseluruhan hasil pengujian dengan *dataset* batik cetak memiliki akurasi yang lebih tinggi (72.86%) dibanding hasil pengujian dengan *dataset* batik tulis (47.62%). Hal ini membuktikan bahwa pengujian dengan *dataset* batik tulis hanya dapat menghasilkan akurasi yang rendah karena batik tulis tidak dapat memiliki pola tekstur.

Tabel 7 Akurasi hasil pengujian batik cetak

Ukuran GLCM	Akurasi keseluruhan
4x4	61.43%
8x8	72.86%
16x16	67.14%

5. KESIMPULAN

Dari hasil perancangan dan pembuatan aplikasi, dapat diambil kesimpulan antara lain:

- Batik tulis tidak memiliki pola berupa tekstur yang dapat diambil dengan GLCM. Hal ini disimpulkan dengan membandingkan hasil pengujian dengan *dataset* batik tulis dan hasil pengujian dengan *dataset* batik cetak. Bahkan dengan menggunakan gambar yang berisi *isen* saja untuk membantu proses *training*, hasil pengujian hanya mencapai hasil maksimal 47.62%
- *Fuzzy* ID3 dapat menghasilkan *decision tree* yang baik meskipun dengan atribut bersifat numerik dalam jumlah besar. *Decision tree* yang dihasilkan dapat digunakan untuk mengenali pola pada data. Dengan jumlah atribut yang besar, program dapat menganalisa atribut mana yang memiliki pengaruh paling besar terhadap hasil secara otomatis, sehingga pengguna tidak perlu memilih secara manual atribut mana saja yang digunakan.

- Secara keseluruhan, ukuran GLCM 8x8 akan memberikan hasil yang terbaik untuk mendapatkan fitur tekstur dari gambar batik. Hal ini dapat dilihat dari hasil pengujian pada subbab 5.3, yaitu hasil pengenalan tertinggi selalu menggunakan ukuran GLCM 8x8.

6. DAFTAR PUSTAKA

[1] Bhatt, H., Mehta, S., D'mello, L.R. 2015. *Use of ID3 Decision Tree Algorithm for Placement Prediction*. International Journal of Computer Science and Information Technologies.

[2] Hall-Beyer, Mryka. *GLCM Texture Tutorial*. http://www.fp.ucalgary.ca/mhallbey/tutorial.htm, diakses 25 November 2015

[3] Intan, R., Yuliana, O.Y. 2009. *Fuzzy Decision Tree Induction Approach for Mining Fuzzy Association Rules*. ResearchGate.

[4] Jiawei, H. 2001. Data Mining: Concepts and Techniques. San Fransisco.

[5] Kemdikbud, Kamus Besar Bahasa Indonesia. http://kbbi.web.id/batik, diakses 25 November 2015

[6] Nurhaida, Ida, et al., 2012. *Performance Comparison Analysis Features Extraction Methods for Batik Recognition*

[7] Pathak, B., Barooah, D. 2013. *Texture Analysis Based On the Gray-Level Co-occurrence Matrix Considering Possible Orientations*, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.

[8] Pratiwi, M., Alexander, Harefa, J., Nanda, S. 2015. *Mammograms Clasification using Gray-level Co-occurrence Matrix and Radial Basis Function Neural Network*. 2015 International Conference on Computer Science and Computational Intelligence.

[9] Quinlan, J.R. 1985. *Induction of Decision Trees*. Boston: Kluwer Academic Publishers.

[10] Santoni, M.M., Sensuse, D.I., Arymurthy, A.M., Fanany, M.I. 2015. *Cattle Race Classification Using Gray Level Co-occurrence Matrix Convolutional Neural Networks*. 2015 International Conference on Computer Science and Computational Intelligence.

[11] Sun, F., Liu, Y., Xurigan, S., Zhang, Q. 2015. *Research of Clothing Sales Prediction Analysis Based on ID3 Decision Tree Algorithm*. International Symposium on Computers & Informatics.

[12] UNESCO. *Fourth Session of the Intergovernmental Committee (4.COM)*. http://www.unesco.org/culture/ich/en/RL/indonesian-batik-00170, diakses 25 November 2015