6. Feature Extraction[4]

The segmentation process is followed by the feature extraction phase. The present research work has primarily used a combination of two feature detectors: the Maximally Stable Extremal Regions (MSER) [131] which is used to detect blob features and then the Speeded Up Robust Features (SURF) [132] which is used to extract the scale-independent features suitable for classification.

6.1 Maximally Stable Extermal Regions (MSER)

Let $Q_1, \ldots, Q_i, \ldots\infty$ be a sequence of nested extremal regions ($Q_{i-1} \subset Q_i$). Extremal region Q_{i*} is maximally stable iff $q(i) = |Q_{i+\Delta} \backslash Q_{i-\Delta}|/|Q_i|$ has a local minima at i*. The extermal regions R_i are detected as follows [133]:

$\forall p \in R_i, \forall q \in \text{boundary}(R_i) \Rightarrow I_{in}(p) >= I_{in}(q)$ …Equation 6.1.1

Where I_{in} is the input image, ***iff*** means ***if and only if***. Here, $Q_1, \ldots, Q_i, \ldots\infty$ is an infinite sequence (ordered set) of nested extremal regions that belong to or forming the outer structure of any image or its parts.

In $|\mathbf{Q_{i+\Delta} \backslash Q_{i-\Delta}}|/|\mathbf{Q_i}|$, $Q_{i+\Delta} \backslash Q_{i-\Delta}$ => the relative complement of $Q_{i+\Delta}$ and $Q_{i-\Delta}$

$|Q_{i+\Delta} \backslash Q_{i-\Delta}|$ and $|Q_i|$ => cardinality of the expression within

$\mathbf{|Q_{i+\Delta} \backslash Q_{i-\Delta}|/|Q_i|}$ => ratio of two absolute values

6.2 Speeded Up Robust Features (SURF)

The sum of the original image within a rectangle can be evaluated quickly using the integral image, $S(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(x,y)$ …Equation 6.2.1

A Hessian matrix H (p, σ) at point p and scale σ, is determined to obtain the point of interest:

$$H(p, \sigma) = \begin{bmatrix} Lxx(p,\sigma) & Lxy(p,\sigma) \\ Lyx(p,\sigma) & Lyy(p,\sigma) \end{bmatrix}$$ …Equation 6.2.2

The lowest level of the scale space is obtained as:

σ_{approx} = current filter size X (base filter scale/base filter size) … Equation 6.2.3

Then, the Gaussian second-order derivative with a box filter is approximated and later, the maxima of the determinant of the Hessian matrix is extracted as candidate interest

[4]Based on author's publication no. 1 and no. 5[Appendix D]

points by using equations 6.2.2 and 6.2.3:

$\text{Det}(H_{approx}) = D_{xx}D_{yy} - (\beta D_{xy})^2$ …Equation 6.2.4

While computing the determinant of the Hessian matrix, the term β compensates for the error caused by the approximation of the true Gaussian derivative masks. An analogous computational form, performed over the entries of a matrix $\sqrt{\sum_{i,j} a_{ij}^2}$ is called the Frobenius norm. The multiplicative factor indicated in the equation 6.2.4 defines the ratio of Frobenius norms between the box filter approximations and their true peers. It has been observed that β may practically be approximated using a static constant 0.9 even though β is not independent of the scale size σ.

6.3 MSER-SURF Methodology

The combined MSER-SURF algorithm may be written as [134]:

Step 1: De-noised image used as input;

Step 2: Extermal regions are detected using MSER (by using equation 6.1.1);

Step 3: Candidate interest points are determined using SURF (by using equation 6.2.4);

Step 4: Finally feature vectors are extracted and merged accordingly:

$\overrightarrow{F_{MSER-SURF}} = \{\overrightarrow{F_{MSER}}, \overrightarrow{F_{SURF}}\}$ … Equation 6.3.1

Equation 6.3.1 implies that SURF descriptors are extracted at locations identified by the MSER detector. Upon analyzing the algorithm, it may be observed that at first the MSER detector incrementally walked through the intensity range of the concerned image to detect stable regions (equation 6.1.1), and then the SURF points of interest were detected from those regions (equation 6.2.4). From each concerned image 12 MSER regions were detected (Figure 33) and a sample of the structure of such a region is as follows: [Count, Location, Orientation, PixelList] = (1, 44.5362, 12.7516,-1.4958, 69*2 int32). This is a typical MSER region structure culled out of an image. The structure consists of a *Count* (no. of stored regions; type integer), *Location* (locations of ellipses, stored as an *M*-by-2 array of [*x y*] coordinates), and *Orientation* (ellipse orientation, stored as a value in the range from -pi/2 to + pi/2 radians. This value represents the orientation of the ellipse as measured from the *X*-axis to the major axis of the ellipse) and *PixelList* (point coordinates

for detected MSER regions, specified as an *M*-by-1 cell array. Each cell is an array of (x, y) coordinates of dimension p-by-2, where p is the number of pixels in an MSER region detected; data type is a 32-bit integer). From these regions, SURF features are retrieved. A neighborhood of size 20s was taken around each of valid SURF points where s is the window size. Each region is divided into 4x4 sub-regions. For each sub-region, horizontal and vertical wavelet-based responses are recorded. This when represented as a vector gave SURF feature descriptor with a total of 64 dimensions. Features are given arbitrary names f_i, i=1 to 64.

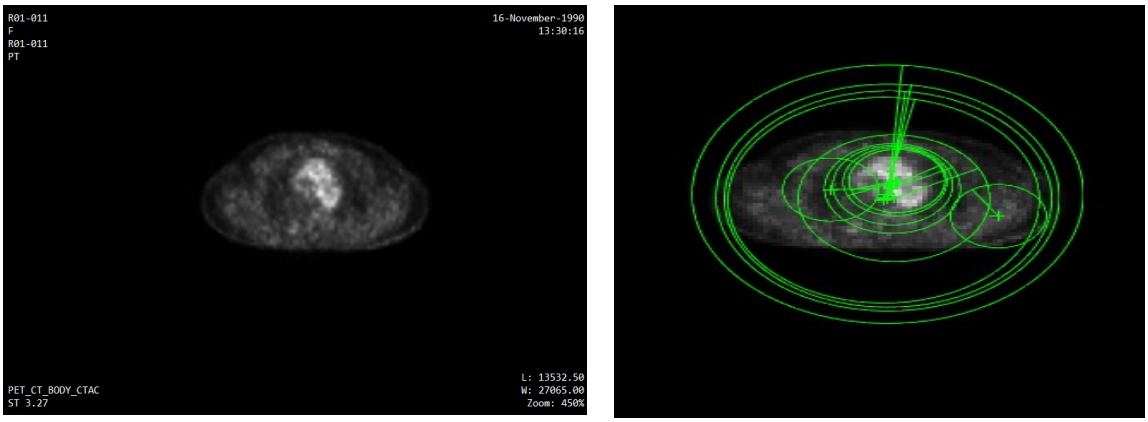

Figure 33: (a) base image (b) image with MSER regions and SURF points detected on it

f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
0.578774	0.341802	0.907862	0.946885	-0.61278	-0.1207	-0.07568	0.224436	0.875566	0.654432	0.407034	0.502841	2.105008	2.93241	1.869117	2.517305	-0.65664	1.032748	0.784053	0.897851
-0.10001	-0.63247	-0.05705	0.209487	1.134052	0.136475	0.704956	0.026082	-0.39125	0.148589	0.200012	0.027693	0.323856	-0.32468	-0.02525	-0.01429	0.019264	-0.08957	-0.21139	-0.1889
0.467441	-0.48853	0.03694	-0.1898	-0.25864	-0.19164	-0.43828	-0.29827	0.28435	0.880797	0.085575	0.721569	-0.22172	0.28327	-0.14555	-0.00686	-0.02017	-0.70893	-0.30386	-0.38381
-0.03201	0.065766	-0.37844	-0.37765	1.99261	2.399467	1.726054	2.221893	-0.7808	1.93831	1.300851	1.828556	-0.98552	0.581482	0.656336	0.269106	0.682647	0.949762	0.415279	0.052271
2.260805	1.390013	1.75832	0.90859	-0.52302	-0.40589	-0.08704	-0.31317	0.085178	-0.40092	-0.13928	-0.54452	-0.58586	0.140184	0.730167	-0.02735	-0.67807	-0.55947	0.611199	0.961426
-0.11817	-0.02085	-0.51321	-0.53583	-0.09623	1.392135	-0.22594	1.242198	-0.455	0.254093	-0.15339	0.05523	0.101633	-0.19568	-0.48503	-0.50075	1.341887	-0.29707	1.062489	0.361841
0.120809	0.351224	-0.29281	-0.15961	-0.63115	-0.14608	-0.6443	-0.25884	-0.00088	-0.45556	-0.59862	-0.68703	0.068654	-0.1915	-0.4462	-0.4636	0.157769	-0.34948	-0.13343	-0.47265
-0.10911	-0.05356	-0.50164	-0.56412	-0.72499	-0.69468	-0.86456	-0.79484	0.10987	-0.66651	-0.71928	-0.89464	0.121453	-0.21239	-0.48076	-0.49344	-0.45482	-0.30236	-0.72242	-0.68209
-0.12479	-0.18875	-0.46834	-0.48078	0.49783	-0.09496	0.144751	-0.19492	0.962377	2.110545	0.873837	2.009904	-1.02731	1.36939	0.70172	0.968683	-0.43797	-0.06619	-0.71423	-0.6924
-0.02093	0.384362	-0.39535	-0.11667	2.596569	1.111917	2.294274	0.967282	-1.65684	1.669529	1.06216	1.545554	-0.5212	0.181743	0.16886	-0.14682	0.003229	-0.23554	-0.28811	0.119497
-0.64425	-0.57761	1.301347	0.934338	1.523284	0.900813	1.291683	1.173476	-0.5252	0.028622	0.360416	0.229731	0.036153	-0.65659	0.638576	0.17524	-0.77184	1.147281	0.080386	0.562724
0.541997	1.932077	3.243887	2.385588	0.936639	0.496033	0.606426	0.506562	2.030926	-0.75438	3.399623	0.362257	1.437835	-3.09231	1.601881	2.379514	0.848948	-0.74682	2.038314	1.165616
0.010111	0.355443	-0.03487	0.140942	1.72442	0.77964	1.206458	0.720517	-0.33095	0.039493	0.309576	0.072036	-0.2481	-0.58393	-0.02289	0.202156	0.507489	-0.2757	0.585233	0.045416
-0.13257	-0.03658	-0.52649	-0.55173	2.421225	0.694337	3.114052	0.558719	0.957306	0.270918	1.02765	0.32021	-3.52191	0.600876	3.340439	1.016578	0.080707	-0.10457	-0.21126	-0.51541
0.073657	-0.47893	-0.2862	-0.17836	-0.80335	-0.75063	-0.72632	-0.69894	0.088762	-0.50723	-0.55737	-0.72914	0.409488	0.245729	-0.13694	-0.08669	-0.38801	-0.63127	-0.64886	-0.35421
0.216038	-0.05243	-0.13751	-0.31743	-0.67436	-0.80605	-0.69963	-0.81137	0.15365	-0.8451	-0.53729	-0.75241	0.104661	-0.24884	-0.32498	-0.34949	-0.18103	0.263383	-0.22822	-0.42601
0.543694	0.231538	0.999222	0.97935	1.48568	1.63953	1.446042	1.503461	-1.30061	1.167117	0.930873	1.180111	2.682846	-0.86369	2.297976	0.552708	1.947328	1.356899	1.673964	1.23045
1.209054	1.079489	0.787083	0.637701	0.697819	4.151942	0.666225	3.9302	-0.71766	2.652664	0.425412	2.580705	-0.82963	0.462092	0.49442	0.120551	6.324245	2.192857	6.097446	2.093466
-0.21031	-0.02177	-0.43858	-0.39612	0.580587	2.432248	0.811212	2.254612	-0.46708	1.826536	0.585048	1.710869	-0.30171	0.238924	-0.05892	-0.0913	0.287648	0.735748	0.001786	-0.10547
3.584811	0.742977	2.902291	1.483609	-0.79728	0.786023	1.17818	0.891787	-0.01397	0.226591	0.355006	0.212877	-0.84093	1.030641	0.518065	0.691103	-0.19625	-0.10162	-0.10387	0.609002
1.480837	0.85516	0.981762	0.952231	0.416319	1.117993	0.278139	0.974128	0.40856	1.062523	1.205364	1.516175	-1.357	1.114449	1.121816	0.747981	0.4657	-0.23268	1.017643	1.786934
-0.05191	-0.10832	-0.35944	-0.45259	2.080528	0.754759	1.477703	0.620517	0.235592	0.503106	0.889871	0.625762	-0.6358	-0.26752	0.33584	-0.04797	0.201447	0.426633	-0.00805	-0.24875
-0.2219	-0.13158	-0.50426	-0.50901	-0.46239	-0.57307	-0.68611	-0.66349	0.100355	-0.12027	-0.5474	-0.33894	0.093872	-0.15744	-0.47345	-0.4571	-0.35215	-0.03068	-0.64665	-0.66948
0.056657	0.107733	-0.34257	-0.33379	-0.60698	-0.21719	-0.66997	-0.31765	0.09762	-0.23288	-0.47457	-0.45344	0.065928	-0.08181	-0.31752	-0.36222	0.111657	0.46084	-0.14511	0.056005

Figure 34: MSER-SURF features extracted from the image collection

6.4 Other Features

The present study has also considered other features including first order and higher-order statistical derivatives. This includes texture-based features portrayed by the Gray Level Co-occurrence Matrix (GLCM) [135]. The present research work has also used other local feature extraction techniques to retrieve local patterns from the concerned images. Major features used in the study apart from MSER-SURF, are as follows:

6.4.1 Entropy

Entropy measures the lost image information during transmission. This is an important feature for loss calculation during image transformation.

$$\text{Entropy} = \sum_{i,j=0}^{N-1} -\ln(P_{ij})\, P_{ij} \quad \text{… Equation 6.4.1.1}$$

6.4.2 Energy

Energy or Angular Second Moment (ASM) is the measure of uniformity in pixels of an image.

$$\text{Energy} = \sum_{i,j=0}^{N-1} (P_{ij})^2 \quad \text{… Equation 6.4.2.1}$$

Where, P_{ij} = $(i,j)^{th}$ pixel of the concerned image and N = number of gray levels in the image

6.4.3 Homogeneity

Local homogeneity is measured by Inverse Different Moment (IDM) depicting if the gray level is uniform.

$$\text{Homogeneity} = \sum_{i,j=0}^{N-1} \frac{P_{ij}}{1+(i-j)^2} \quad \text{… Equation 6.4.3.1}$$

6.4.4 Correlation

Correlation shows in an image how much correlated a pixel location is with its neighboring pixels.

$$\text{Correlation} = \sum_{i,j=0}^{N-1} P_{ij} \frac{(i-\mu)(j-\mu)}{\sigma^2} \quad \text{… Equation 6.4.4.1}$$

6.4.5 Contrast

It calculates the intensity contrast between a pixel and its adjacent pixels across the input image.

$$\text{Contrast} = \sum_{i,j=0}^{N-1} P_{ij}(i-j)^2 \quad \text{…Equation 6.4.5.1}$$

6.4.6 Mean

The GLCM mean is the mean intensity of pixels, $\mu = \sum_{i,j=0}^{N-1} iP_{ij}$ …Equation 6.4.6.1

6.4.7 Variance

The calculation of the variance of the intensities of all candidate pixel locations in the set of relationships that contribute to the GLCM:

$$\sigma^2 = \sum_{i,j=0}^{N-1} P_{ij}(i-\mu)^2 \quad \text{…Equation 6.4.7.1}$$

mean	sd	entropy	centroidx	centroidy	area	eccent	solidity	contrastx	contrasty	homox	homoy	correlx	correly	energx	energy
0.006897	0.082764	0.059435	79	67.70796	113	0.818106	0.957627	0.002728	0.003472	0.998636	0.998264	0.803936	0.750464	0.983365	0.982625
0.010132	0.100149	0.081666	73.44578	49.63855	166	0.853121	0.892473	0.004216	0.004216	0.997892	0.997892	0.793051	0.793051	0.975428	0.975428
0.023682	0.15206	0.161641	65.54897	64.35567	388	0.511559	0.97	0.00496	0.005704	0.99752	0.997148	0.894366	0.878521	0.948107	0.947371
0.004578	0.067505	0.042163	68.76	73.88	75	0.685435	0.9375	0.002232	0.00248	0.998884	0.99876	0.758879	0.732087	0.988515	0.988269
0.018066	0.133196	0.130442	62.19257	57.47297	296	0.342556	0.951768	0.004712	0.004712	0.997644	0.997644	0.869221	0.869221	0.959277	0.959277
0.023193	0.150522	0.159013	68.28421	80.12105	380	0.654716	0.938272	0.0062	0.005332	0.9969	0.997334	0.865246	0.884112	0.947825	0.948683
0.00885	0.093661	0.07307	65.57931	64.36552	145	0.690578	0.917722	0.003472	0.003472	0.998264	0.998264	0.805145	0.805145	0.97872	0.97872
0.020996	0.143375	0.146997	78.29942	57.19477	344	0.715253	0.828916	0.008185	0.008929	0.995908	0.995536	0.803958	0.786136	0.950134	0.949402
0.016235	0.126383	0.119746	62.38722	74.08271	266	0.846886	0.93662	0.005704	0.00372	0.997148	0.99814	0.824168	0.885327	0.961886	0.963852
0.010498	0.101924	0.084077	73	56.53488	172	0.827223	0.934783	0.003472	0.004216	0.998264	0.997892	0.835454	0.800195	0.975438	0.9747
0.017212	0.130064	0.125486	82.93617	72.71986	282	0.727651	0.886792	0.005704	0.0062	0.997148	0.9969	0.833976	0.81954	0.959969	0.959479
0.003418	0.058365	0.032925	71.5	86.41071	56	0.581167	1	0.001984	0.001984	0.999008	0.999008	0.71329	0.71329	0.991099	0.991099
0.010254	0.100744	0.082472	74.42857	60.93452	168	0.590845	0.879581	0.00434	0.003968	0.99783	0.998016	0.789474	0.807519	0.975062	0.975431
0.007996	0.089063	0.067191	85.09924	55.9313	131	0.75633	0.942446	0.002852	0.00372	0.998574	0.99814	0.82299	0.769117	0.981043	0.980181
0.024597	0.154899	0.166527	62.58065	65.933	403	0.707192	0.948235	0.00496	0.0062	0.99752	0.9969	0.898201	0.872751	0.946338	0.945112
0.006531	0.080551	0.056795	66.30841	66.75701	107	0.770868	0.955357	0.00248	0.003224	0.99876	0.998388	0.811836	0.755386	0.984345	0.983605
0.008911	0.09398	0.073485	65.5411	67.24658	146	0.701759	0.960526	0.00372	0.002976	0.99814	0.998512	0.792643	0.834115	0.978352	0.979091
0.004639	0.067952	0.042636	65.69737	72.82895	76	0.625433	0.962025	0.002232	0.00248	0.998884	0.99876	0.762037	0.735596	0.988393	0.988146
0.002808	0.052914	0.027843	58.06522	59.93478	46	0.311526	1	0.001736	0.001736	0.999132	0.999132	0.694782	0.694782	0.992579	0.992579
0.007019	0.083488	0.060309	74.63478	61.56522	115	0.779683	0.966387	0.003472	0.00248	0.998264	0.99876	0.754773	0.824838	0.982381	0.983367
0.002794	0.052786	0.027728	114.8544	114.5146	103	0.486528	0.990385	0.001206	0.001096	0.999397	0.999452	0.785803	0.805275	0.993164	0.993274
0.009705	0.098036	0.078828	57.67925	61.18239	159	0.629378	0.97546	0.003472	0.003224	0.998264	0.998388	0.822146	0.83485	0.977017	0.977263
0.007751	0.087703	0.065488	72.11811	62.01575	127	0.608784	0.969466	0.003224	0.002728	0.998388	0.998636	0.793651	0.825397	0.981161	0.981654

Figure 35: GLCM and other primitive features extracted from the image dataset

6.4.8 FAST-HOG Features

To retrieve other local features like Binary Large Objects (BLOBs), corners, and edge pixels, etc., a combination of Features from Accelerated Segment Test (FAST) [136] and Histogram of Oriented Gradients (HOG) [137] methods have been used in the present research work. Local features refer to a distinct pattern found in an image. The pattern differs by texture, color, or intensity from its conjugated vicinity. FAST features were detected and the strongest corners are selected. As FAST features are not much suitable for classification, the HOG features are extracted from the selected corners along with valid points from the detected features (Figure 36).

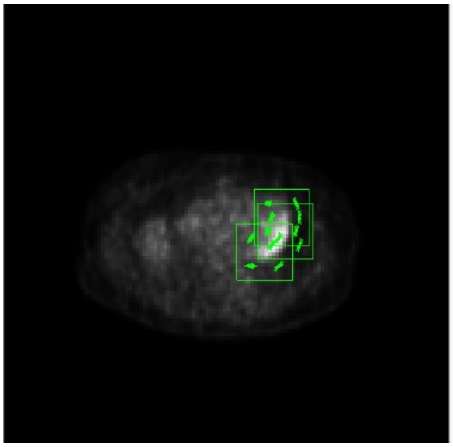

Figure 36: FAST-HOG features detected on NSCLC tumor

f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
0.578774	0.341802	0.907862	0.946885	-0.61278	-0.1207	-0.07568	0.224436	0.875566	0.654432	0.407034	0.502841	2.105008	2.93241	1.869117	2.517305	-0.65664	1.032748	0.784053	0.897851
-0.10001	-0.63247	-0.05705	0.209487	1.134052	0.136475	0.704956	0.026082	-0.39125	0.148589	0.200012	0.027693	0.323856	-0.32468	-0.02525	-0.01429	0.019264	-0.08957	-0.21139	-0.1889
0.467441	-0.48853	0.03694	-0.1898	-0.25864	-0.19164	-0.43828	-0.29827	0.28435	0.880797	0.085575	0.721569	-0.22172	0.28327	-0.14555	-0.00686	-0.02017	-0.70893	-0.30386	-0.38381
-0.03201	0.065766	-0.37844	-0.37765	1.99261	2.399467	1.726054	2.221893	-0.7808	1.93831	1.300851	1.828556	-0.98552	0.581482	0.656336	0.269106	0.682647	0.949762	0.415279	0.052271
2.260805	1.390013	1.75832	0.90859	-0.52302	-0.40589	-0.08704	-0.31317	0.085178	-0.40092	-0.13928	-0.54452	-0.58586	0.140184	0.730167	-0.02735	-0.67807	-0.55947	0.611199	0.961426
-0.11817	-0.02085	-0.51321	-0.53583	-0.09623	1.392135	-0.22594	1.242198	-0.455	0.254093	-0.15339	0.05523	0.101633	-0.19568	-0.48503	-0.50075	1.341887	-0.29707	1.062489	0.361841
0.120809	0.351224	-0.29281	-0.15961	-0.63115	-0.14608	-0.6443	-0.25884	-0.00088	-0.45556	-0.59862	-0.68703	0.068654	-0.1915	-0.4462	-0.4636	0.157769	-0.34948	-0.13343	-0.47265
-0.10911	-0.05356	-0.50164	-0.56412	-0.72499	-0.69468	-0.86456	-0.79484	0.10987	-0.66651	-0.71928	-0.89464	0.121453	-0.21239	-0.48076	-0.49344	-0.45482	-0.30236	-0.72242	-0.68209
-0.12479	-0.18875	-0.46834	-0.48078	0.49783	-0.09496	0.144751	-0.19492	0.962377	2.110545	0.873837	2.009904	-1.02731	1.36939	0.70172	0.968683	-0.43797	-0.06619	-0.71423	-0.6924
-0.02093	0.384362	-0.39535	-0.11667	2.596569	1.111917	2.294274	0.967282	-1.65684	1.669529	1.06216	1.545554	-0.5212	0.181743	0.16886	-0.14682	0.003229	-0.23554	-0.28811	0.119497
-0.64425	-0.57761	1.301347	0.934338	1.523284	0.900813	1.291683	1.173476	-0.5252	0.028622	0.360416	0.229731	0.036153	-0.65659	0.638576	0.17524	-0.77184	1.147281	0.080386	0.562724
0.541997	1.932077	3.243887	2.385588	0.936639	0.496033	0.606426	0.506562	2.030926	-0.75438	3.399623	0.362257	1.437835	-3.09231	1.601881	2.379514	0.848948	-0.74682	2.038314	1.165616
0.010111	0.355443	-0.03487	0.140942	1.72442	0.77964	1.206458	0.720517	-0.33095	0.039493	0.309576	0.072036	-0.2481	-0.58393	-0.02289	0.202156	0.507489	-0.2757	0.585233	0.045416
-0.13257	-0.03658	-0.52649	-0.55173	2.421225	0.694337	3.114052	0.558719	0.957306	0.270918	1.02765	0.32021	-3.52191	0.600876	3.340439	1.016578	0.080707	-0.10457	-0.21126	-0.51541
0.073657	-0.47893	-0.2862	-0.17836	-0.80335	-0.75063	-0.72632	-0.69894	0.088762	-0.50723	-0.55737	-0.72914	0.409488	0.245729	-0.13694	-0.08669	-0.38801	-0.63127	-0.64886	-0.35421
0.216038	-0.05243	-0.13751	-0.31743	-0.67436	-0.80605	-0.69963	-0.81137	0.15365	-0.8451	-0.53729	-0.75241	0.104661	-0.24884	-0.32498	-0.34949	-0.18103	0.263383	-0.22822	-0.42601
0.543694	0.231538	0.999222	0.97935	1.48568	1.63953	1.446042	1.503461	-1.30061	1.167117	0.930873	1.180111	2.682846	-0.86369	2.297976	0.552708	1.947328	1.356899	1.673964	1.23045
1.209054	1.079489	0.787083	0.637701	0.697819	4.151942	0.666225	3.9302	-0.71766	2.652664	0.425412	2.580705	-0.82963	0.462092	0.49442	0.120551	6.324245	2.192857	6.097446	2.093466
-0.21031	-0.02177	-0.43858	-0.39612	0.580587	2.432248	0.811212	2.254612	-0.46708	1.826536	0.585048	1.710869	-0.30171	0.238924	-0.05892	-0.0913	0.287648	0.735748	0.001786	-0.10547
3.584811	0.742977	2.902291	1.483609	-0.79728	0.786023	1.17818	0.891787	-0.01397	0.226591	0.355006	0.212877	-0.84093	1.030641	0.518065	0.691103	-0.19625	-0.10162	-0.10387	0.609002
1.480837	0.85516	0.981762	0.952231	0.416319	1.117993	0.278139	0.974128	0.40856	1.062523	1.205364	1.516175	-1.357	1.114449	1.121816	0.747981	0.4657	-0.23268	1.017643	1.786934
-0.05191	-0.10832	-0.35944	-0.45259	2.080528	0.754759	1.477703	0.620517	0.235592	0.503106	0.889871	0.625762	-0.6358	-0.26752	0.33584	-0.04797	0.201447	0.426633	-0.00805	-0.24875
-0.2219	-0.13158	-0.50426	-0.50901	-0.46239	-0.57307	-0.68611	-0.66349	0.100355	-0.12027	-0.5474	-0.33894	0.093872	-0.15744	-0.47345	-0.4571	-0.35215	-0.03068	-0.64665	-0.66948
0.056657	0.107733	-0.34257	-0.33379	-0.60698	-0.21719	-0.66997	-0.31765	0.09762	-0.23288	-0.47457	-0.45344	0.065928	-0.08181	-0.31752	-0.36222	0.111657	0.46084	-0.14511	0.056005

Figure 37: Glimpse of FAST-HOG features detected from the image collection

Figure 36 displays the original image with an overlay of HOG features around the strongest corners detected by FAST. HOG features are extracted from grayscale input images. HOG features are extracted around specified point locations. The function returns the locations from the input whose adjacent pixels are fully constrained by the input image. Scale information linked with the points is overlooked. HOG features having length N is returned as a 1-by-N vector, where. Local shape information is being encoded by the returned features (Figure 37). Its various input parameters are as follows:

a) Input image, specified M-by-N 2-D grayscale. The input image should be a real, non-sparse value. Images may lose detailed shape information that the HOG function can encode due to rigid cropping. The issue may be resolved by including padding of background pixels around the cropped area.

b) Center location point formed as an M-by-2 matrix having M number of (x, y) coordinates. Descriptors are being generated from the adjacent points that are fully confined by the image edge. The size of the neighborhood pixels is determined by the *BlockSize* parameter. Pixel locations that exist within the image boundary are often used to decide the valid output points. The function ignores the scale information associated with these points.

c) Cell size is expressed as a vector comprising pixel information. To capture large-scale spatial information, increase cell size. If the cell size is magnified, minute details regarding scale may get lost.

d) The number of cells in a block, specified as a 2-element vector. The ability to suppress local illumination changes may be reduced due to a large block-size value. As the number of pixels exists in a chunk, these changes may get lost with averaging. To capture the significant local pixels block-size may have to be reduced. Smaller block sizes can help suppress illumination changes of HOG features.

e) The number of overlapping cells between adjacent blocks, specified as a 2-element vector. Although large overlap values can confine ample information, they produce a larger feature vector size.

f) The number of orientation histogram bins, specified as a positive scalar. To encode finer orientation details, the number of bins should be increased. The increasing number of bins increases the size of the feature vector, which requires more processing overhead.

g) Selection of orientation values, specified as a logical scalar. If this property becomes true, orientation values get uniformly spaced within a span of bins between -180 and 180 degrees. If this property becomes false, it is evenly distributed from 0 through 180. Signed orientation differentiates between light-to-dark and dark-to-light transitions.

The output arguments are as follows:

a) Extracted features, returned as either a vector or a matrix. The local shape information may be encoded by features retrieved from regions or point locations within an image.

b) Valid points are associated with each feature descriptor vector output. This output can be returned as an M-by-2 matrix of (x,y) coordinates. The function culls out M number of attributes from candidate points in a region having a size equal to [CellSize.*BlockSize]. The extracted descriptors are the same type of object or matrix as that of the input.

6.5 Feature Selection

Feature selection or dimensionality reduction is a very important step in data analysis where the strongest spatial features are retained for the study from a pile of features. Normally the feature selection is done by running a Principal Components Analysis (PCA) [138] for dimensionality reduction. This is typically accomplished by selecting enough eigenvectors to account for 95% of the variance in the extracted feature set. In the present study, feature selection or dimensionality reduction has been done by Independent Component Analysis (ICA) [139] which is more advanced than PCA. PCA optimizes the second-order statistics and finds uncorrelated components, whereas ICA optimizes higher-order statistics and finds independent components [140]

6.5.1 Algorithm

If $X=[X_1, X_2 \ldots X_m]^T$ is a random observed vector having m elements which are a mix of m independent elements of another random vector $S= [S_1, S_2, \ldots, S_m]^T$, then X may be expressed as:

$$X=AS \qquad \ldots\text{Equation 6.5.1.1}$$

Where A is an m-by-m mixing matrix.

The goal of ICA is to estimate A and also estimate the source distribution S. Methods like PCA which is based on second-order moments, cannot recover A. ICA uses higher-order moments to recover A.

Now, pre-assumptions made during ICA are as follows:

a) sources are statistically independent;
b) mixing matrix is square;
c) there should be no external noise;
d) data have zero mean;
e) signals must not have a normal probability density

Statistical independence may be expressed as:

$E[g_1(x_i)\, g_2(x_j)] - E[g_1(x_i)]\, E[g_2(x_j)] = 0$ for $i \neq j$ …Equation 6.5.1.2

Where E[.] is the expectation for any function g(x). In specific, Gaussian uncorrelatedness is equivalent to independence. In ICA typically Minimization of mutual information or Maximization of non-Gaussianity is done. Independence may be achieved by forcing two random variables to be as far from the normal distribution as possible by measuring the non-gaussianity. Negentropy or the positive measure of gaussianity.are approximated to calculate the non-gaussianity. This algorithm is called FastICA:

1. *Center S by subtracting the mean*
2. *Whiten S*
3. *Choose a random initial value w for the de-mixing matrix*
4. *Calculate the new value for A*
5. *Normalize A*
6. *Check if the algorithm has converged and if it hasn't, resume from step 4*
7. *Take the dot product of A and S to get the independent source signals X (Equation 6.5.1.1)*

The eigenvalues are being decomposed w.r.t. its covariance matrix to whitening a signal. The potential correlations between the components of a signal are removed. This may be expressed as:

$$\tilde{x} = ED^{-1/2}E^{T}x$$ …Equation 6.5.1.3

Where D is a diagonal matrix of eigenvalues (lambda).

6.5.2 Methodology

In the present research work, transformation on numeric data has been done using the FastICA algorithm [141]. First, the data whitening (decoupling transform) has been done. Then, the FastICA main loop has been executed for 200 iterations with an error tolerance value 1.0E-4 (for solution convergence). Strongest 64 features are taken for final classification based on their respective ranks in ICA analysis.

f1	f2	f3	f4	f5	f6	f7	...	f58	f59	f60	f61	f62	f63	f64
0.439602	0.547821	0.00481	0.006076	0.145296	0.215474	0.04261	...	0.623217	0.314782	0.314895	0.359191	0.495266	0.116026	0.118451
0.451386	0.546407	0.037355	0.01083	0.130287	0.206117	0.049503	...	0.808204	0.029337	0.022213	0.435111	0.565142	0.011436	0.043668
0.425059	0.642745	0.220604	0.48215	0.200386	0.442791	0.204353	...	0.644277	0.351163	0.257528	0.268111	0.464487	0.263158	0.164063
0.451386	0.546407	0.037355	0.01083	0.130287	0.206117	0.049503	...	0.808204	0.029337	0.022213	0.435111	0.565142	0.011436	0.043668
0.464168	0.496577	0.051501	0.103768	0.103694	0.158244	0.071917	...	0.403364	0.265747	0.503787	0.391405	0.49696	0.064354	0.109445
0.441988	0.550917	0.009033	0.011411	0.400033	0.364758	0.266112	...	0.745275	0.141209	0.161274	0.628167	0	0.39599	1
0.440039	0.548542	0.005582	0.007051	0.127487	0.202639	0.03426	...	0.505227	0.198528	0.38514	0.333822	0.534269	0.160191	0.104016
0.44191	0.548837	0.011732	0.017664	0.125539	0.177816	0.025069	...	0.82816	0.003934	0.003029	0.429907	0.560158	0.002133	0.00317
0.434875	0.558347	0.050841	0.032015	0.134413	0.202589	0.051069	...	0.771489	0.114443	0.15947	0.394151	0.521096	0.067847	0.106775
0.439602	0.547821	0.00481	0.006076	0.145296	0.215474	0.04261	...	0.623217	0.314782	0.314895	0.359191	0.495266	0.116026	0.118451
0.434875	0.558347	0.050841	0.032015	0.134413	0.202589	0.051069	...	0.771489	0.114443	0.15947	0.394151	0.521096	0.067847	0.106775
0.440039	0.548542	0.005582	0.007051	0.127487	0.202639	0.03426	...	0.505227	0.198528	0.38514	0.333822	0.534269	0.160191	0.104016
0.441988	0.550917	0.009033	0.011411	0.400033	0.364758	0.266112	...	0.745275	0.141209	0.161274	0.628167	0	0.39599	1
0.438737	0.554547	0.003278	0.018076	0.133879	0.223408	0.038016	...	0.11798	0.087388	0.841907	0.304226	0.006421	0.204818	0.938166
0.430309	0.570249	0.019813	0.047477	0.126579	0.231136	0.036686	...	0.798131	0.012771	0.038544	0.425244	0.54657	0.009323	0.025348
0.455201	0.542289	0.037457	0.025837	0.133796	0.157754	0.062879	...	0.829217	0.028993	0.011555	0.435327	0.552661	0.015483	0.017582
0.434875	0.558347	0.050841	0.032015	0.134413	0.202589	0.051069	...	0.771489	0.114443	0.15947	0.394151	0.521096	0.067847	0.106775
0.430309	0.570249	0.019813	0.047477	0.126579	0.231136	0.036686	...	0.798131	0.012771	0.038544	0.425244	0.54657	0.009323	0.025348
0.440039	0.548542	0.005582	0.007051	0.127487	0.202639	0.03426	...	0.505227	0.198528	0.38514	0.333822	0.534269	0.160191	0.104016
0.438737	0.554547	0.003278	0.018076	0.133879	0.223408	0.038016	...	0.11798	0.087388	0.841907	0.304226	0.006421	0.204818	0.938166
0.441025	0.553973	0.007328	0.017022	0.118203	0.205648	0.026126	...	0.765321	0.032049	0.07289	0.449336	0.51297	0.051154	0.082131
0.455345	0.561656	0.034112	0.042714	0.115512	0.270605	0.049775	...	0.765962	0.092994	0.129534	0.308909	0.503025	0.197252	0.098937
0.440648	0.575651	0.037835	0.060644	0.180319	0.623649	0.361021	...	0.432169	0.177026	0.46923	0.370885	0.447306	0.102363	0.196295
0.470039	0.607865	0.067003	0.143087	0.354453	0.87161	0.584796	...	0.389561	0.141565	0.528585	0.414075	0.545626	0.038118	0.044827

Figure 38: Selected features as a result of ICA

6.6 Conclusion

Feature extraction is an unavoidable step in the orthodox machine learning curriculum. Although the manual feature extraction causes processing overhead, it is necessary to feed features in the traditional machine learning algorithms as they do not have an automatic way of extracting features. Such features may turn out very handy when limited hardware resources are there and images cannot be automatically processed. In the present study, different BLOB or Corner based feature extraction techniques have been described. MSER-SURF and FAST-HOG based techniques have been used. Other primitive and higher-order features have also been extracted by deriving first and second-order statistical derivatives. Extracted features have been further used for database preparation.